UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Evolução, Avaliação e Validação do *Software* RoboEduc

Renata Pitta Barros

Orientador: Prof. Dr. Aquiles Medeiros Filgueira Burlamaqui

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Natal, RN, Fevereiro de 2011

Evolução, Avaliação e Validação do Software RoboEduc

Renata Pitta Barros

Dissertação de Mestrado aprovada em 18 de fevereiro de 2011 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Aquiles Medeiros Filgueira Burlamaqui (orientador) .... ECT/UFRN

Prof[a] M.Sc. Carmen Ribeiro Faria Santos ..... UFES/UFES

Prof. Dr. Francisco Milton Mendes Neto ..... UFERSA/UFERSA

Prof. Dr. Luiz Marcos Garcia Gonçalves ..... DCA/UFRN

Prof. Dr. Luiz Eduardo Cunha Leite ..... ECT/UFRN

Agradecimentos

Ao meu orientador professor Aquiles e ao professor Luiz Marcos Garcia, sou grata pela orientação e confiança depositados em minha pessoa.

Aos colegas do Laboratório NatalNet em especial ao grupo de pesquisa de robótica pedagógica RoboEduc, pelas críticas e sugestões.

À minha família, meu pai, minha mãe e meu irmão, pelo apoio durante toda esta jornada.

Ao meu namorado Aderson Jamier pelo apoio nos momentos difíceis.

Resumo

Considerando a transição entre a sociedade industrial para a sociedade da informação, percebemos que a formação digital que é abordada atualmente é insuficiente para navegar no interior de uma realidade digitalizada. Como proposta para minimizar este problema, o presente trabalho avalia, valida e evolui o *software* **RoboEduc**, para trabalhar com a robótica educacional tendo como principal diferencial a programação de dispositivos robóticos em níveis, considerando as especificidades da realidade formativa. Uma das ênfases deste trabalho está na apresentação dos procedimentos e materiais que envolveram o desenvolvimento, a análise e a evolução deste *software*. Para sua validação foram realizados testes de usabilidade, baseado na análise destes testes foi desenvolvida a versão 4.0 do **RoboEduc**.

Palavras-chave: Robótica Educacional, Inclusão Digital, Teste de Usabilidade.

Abstract

Considering the transition from industrial society to information society, we realize that the digital training that is addressed is currently insufficient to navigate within a digitized reality. As proposed to minimize this problem, this paper assesses, validates and develops the software **RoboEduc** to work with educational robotics with the main differential programming of robotic devices in levels, considering the specifics of reality training . One of the emphases of this work isthe presentation of materials and procedures involving the development, analysis and evolution of this software. For validation of usability tests were performed, based on analysis of these tests was developed version 4.0 of **RoboEduc**.

Keywords: Educational Robotics, Digital Inclusion, Usability Testing.

Sumário

Lista de Figuras

Lista de Tabelas

Tabelas

Capítulo 1

Introdução

No enfrentamento da transição entre a sociedade industrial e a sociedade da informação, percebemos que a formação digital com vistas para a inclusão digital abordada atualmente é insuficiente para navegar no interior de uma sociedade digitalizada. Surpreendentemente, as próprias condições formativas carecem de alternativas para o desenvolvimento de um novo referencial educacional o qual preze pela formação do sujeito para sua inserção no mundo digital.

Esse processo de transição gerou gradações de acesso à tecnologia. Podemos exemplificar esta afirmação por meio do estudo proposto por Zwierewicz [Vallejo & Zwierewicz 2007] sobre a situação dos sujeitos mediante o surgimento das TIC's (Tecnologias da Informação e Comunicação). Para os autores, é possível dividir a sociedade brasileira em três níveis quanto ao acesso às novas tecnologias da informação. Os níveis consistem em: analfabetos digitais, migrantes e nativos digitais.

Esses grupos têm características peculiares. Segundo [Vallejo & Zwierewicz 2007], os analfabetos digitais são definidos como sujeitos que não dominam as competências e habilidades necessárias para usufruir da tecnologia digital, sendo marginalizados pela rápida evolução tecnológica. Os migrantes digitais caracterizam-se por serem sujeitos que buscam formas de se inserir no mundo digital e que encaram as tecnologias digitais como ferramentas que necessitam dominar. E, por fim, os nativos digitais são sujeitos que nascem em um mundo digital, dominam e usufruem das tecnologias digitais nas diferentes dimensões de suas vidas.

Observamos, portanto, que a incorporação das tecnologias da informação e comunicação na sociedade contemporânea provocam estágios de exclusão/inclusão digital, uma vez que as situações cotidianas nos confirmam esta afirmativa.

Tecendo reflexões sobre as necessidades de uma nova realidade em construção, o Laboratório NatalNet do Departamento de Engenharia de Computação - DCA da Universidade Federal do Rio Grande do Norte - UFRN executa diversos projetos desde de 2006, para contribuir com esta nova realidade social. O projeto Inclusão Digital Usando Robôs e Avatares (RoboEduc) é um deles [RoboEduc 2010].

O projeto [de Informática-UFRN 2010] tem por objetivo o desenvolvimento de uma metodologia inovadora para controle e programação de robôs por alunos dos anos iniciais e realizar experimentos proporcionando peças teatrais e oficinas com uso de kits de robótica em escolas do ensino fundamental e médio, da capital e do interior do estado.

Assim, no decorrer deste trabalho de pesquisa, estaremos propondo uma maneira alternativa de tratar o problema da carência da educação digital (com uso de robôs) que, certamente, pode ser usado para uma inovação metodológica para a multiplicidade das fontes de obtenção de informações e para a diversidade nas formas de aprender por meio da educação digital social com vista a inclusão digital.

Com o intuito de utilizar a robótica educacional como ferramenta para a educação digital principalmente para alunos iniciantes na vida escolar, nesse trabalho foi realizada uma análise sobre as diversos *softwares* disponíveis para tal propósito, como o Scratch [Malan & Leitner 2007], Alice [Wang et al. 2009], RoboLab [Wang & Wang 2001], Imagine [de Abreu et al. 2002], SuperLogo [Chella 2002] e Legal [de Educação Tecnológica da PETe 2008]. Através de experimentos realizados desde 2006 em escolas públicas da periferia de Natal, segundo Aranibar foi possível avaliar que estes *softwares* não favoreciam a educação digital, pois não configuram novas formas de aprender e não inovam o tempo de aprendizagem, pois não levam em consideração as especificidades educacionais de cada indivíduo [Aranibar et al. 2006].

Existem vários *softwares*, como os citados anteriormente (livres e comerciais), para programação de robôs. Porém, todos são voltadas a usuários com conhecimentos mínimos em computação ou com pelo menos os conceitos básicos de robótica, outras têm o objetivo apenas de ensinar uma linguagem de programação.

Como dito anteriormente, o nosso público alvo é heterogêneo em virtude da migração da sociedade industrial para a sociedade da informação, os quais não possuem certos conhecimentos necessários para utilização desses *softwares* como, por exemplo: lógica de programação para utilizar os controles de fluxo e compreender as funcionalidades de baixo nível do robô, como: motores, sensores, entre outros.

A linguagem brickOS [brickOS 2010] e o NQC [NQC 2010], ambas baseadas na linguagem de programação C, não são simples para qualquer pessoa entender, principalmente caso esta não possua intimidade com a tecnologia. O mesmo acontece com a plataforma leJOS [LeJOS 2010], baseado na linguagem de programação Java.

Portanto, em virtude da ausência de um *software* que atenda convenientemente o nosso público, tornou-se necessário o desenvolvimento de um *software* de robótica educacional específico, que permitisse o ensino da robótica utilizando técnicas de aprendizado colaborativo que oferecesse uma maior flexibilidade na linguagem utilizada para ministrar esse conteúdo. O *softwares* que foi proposto denomina-se **RoboEduc** [Barros 2008]. Este *softwares* possui uma linguagem de programação em níveis para ensino de robótica educacional chamada **Educ**.

O presente trabalho visa expor e principalmente avaliar as principais características do projeto do *software*, os procedimentos metodológicos e educacionais desenvolvidos para sua implementação de forma eficiente para que essa *softwares* possa contribuir para uma educação digital dos sujeitos envolvidos.

As contribuições deste trabalho são:

1. Análise de alguns *software* educacionais;
2. Análise das versões do *software* **RoboEduc**;
3. A versão 4.0 do *software* **RoboEduc**, que tem como novas funcionalidades o projetar e o simulador 2D da linguagem **Educ**;

4. Validação da linguagem **Educ** para que possa auxiliar na educação digital contemplando os três níveis da sociedade atual ao mesmo tempo. Assim será possível que os discentes, com diferentes níveis de cognição, possam realizar a mesma atividade proposta pelo docente respeitando as suas limitações e contemplando as suas competências;
5. A migração da linguagem de baixo nível para NXC;
6. A migração do protocolo de comunicação para o *Bluetooth*;
7. Aprimoramento da linguagem **Educ** para suportar a interação com sensores de diversos tipos.

O presente trabalho, além de compilar resultados de anos de desenvolvimento da linguagem, *software* e metodologia **RoboEduc**, ele amplia suas funcionalidades e as apresenta de uma maneira mais completa e objetiva. Trazemos ainda uma avaliação de usabilidade do *software*. Essa avaliação resultou na alteração e construção de uma nova versão do **RoboEduc**. Essa versão é mais adequada ao nosso público alvo e que apresentamos aqui.

1.1 Motivação

O diferencial deste trabalho está em trazer para o Rio Grande do Norte um *software* de robótica tipicamente brasileiro, com linguagem em língua portuguesa e interfaces acessíveis, que ofereça maior poder de criação aos alunos desde a educação infantil ao ensino superior. A partir deste *software* o professor poderá construir suas próprias aulas, de acordo com sua necessidade. Ao aliar este novo *software* à metodologia de ensino, o professor está construindo uma proposta metodológica fundamentada, com sugestões variadas de atividades para diversos níveis de ensino, com objetivos específicos e ricos em elementos lúdicos.

Um dos principais tópicos associados ao ensino da robótica educacional é o ensino da programação. Sabe-se que existem diversas linguagens de programação, mas os *softwares* que acompanham esses kits possuem interfaces complexas com linguagens de programação de uma das duas formas: gráfica ou textual. As interfaces de programação gráfica normalmente são representações em um nível muito próximo ao controle direto de peças de hardware (ex: criança clica na figura do motor A e digita quanto tempo quer que ele fique acionado), o que não é interessante dependendo da faixa etária e do que se deseja trabalhar com a criança através do uso da robótica. Já os kits de robótica que podem ser programados de forma textual, apresentam uma linguagem de programação tão complexa quanto a utilizada por universitários e, normalmente, baseada na língua inglesa.

Os kits mais completos de robótica educacional são acompanhados de metodologias de ensino que aliam a robótica ao aprendizado das outras disciplinas, como informática, matemática, física, biologia, geografia, etc. Logo, a robótica educacional torna-se um aliado poderoso na educação digital. Em contrapartida, o alto custo e a complexidade do aluno em trabalhar com esses kits, muitas vezes não são adequados a certas idades, atuando como barreira e impedindo à difusão dos benefícios da robótica educacional à sociedade.

Por estas deficiências encontradas e pelos benefícios que a ferramenta da robótica educacional possui no ensino, temos por motivação a criação de um kit de robótica. Este kit é composto por um *software* chamado **RoboEduc** e por uma metodologia própria de ensino. Não focamos no desenvolvimento do *hardware* para o presente momento, o *hardware* utilizado é o do kit da Lego, porém nada impede que seja utilizado com qualquer outra plataforma, apenas é necessário que o fabricante do hardware disponibilize a sua API.

1.2 Objetivo

Este trabalho tem como objetivo a avaliação e validação do **RoboEduc 4.0**. Através dos resultados obtidos, propomos melhorias de desenvolvimento e validamos que esse *software* permite a utilização desta ferramenta como auxilio ao professor para educação digital.

Com os resultados dos testes de avaliação propusemos uma nova versão do *software* que possui cinco diferentes níveis de complexidade de programação, onde cada nível tem a sua própria interface e funcionalidades que se expandem de nível para nível.

Estas distinções de níveis foram desenvolvidas no intuito de responder as necessidades educacionais do usuário considerando o seu nível de maturação cognitiva, conforme descrito na nossa metodologia que esta na Seção 1.3. Além disso, existem outras atividades pedagógicas que podem ser trabalhadas assim como outras competências como lateralidade, criatividade e psicomotricidade.

O objetivo desse trabalho está na apresentação da importância do processo de realização dos testes de usabilidade no *software* **RoboEduc**. Para isto foram propostos os procedimentos, métodos, roteiros e materias que envolveram a realização e condução dos testes assim como análise dos resultados do *software* **RoboEduc**, principalmente da sua linguagem **Educ**.

1.3 Metodologia

A metodologia utilizada para a elaboração dos testes de usabilidade, que é um dos focos desse trabalho, tem como base os conceitos de usabilidade guiados pela área de IHC (Interação Homem-Computador). Os testes de usabilidade se tornam mais eficientes quando realizados ao longo do ciclo de vida do desenvolvimento do *software*. Portanto, os vários tipos de testes permitem identificar se há alguma deficiência e, também, buscam melhorar a usabilidade do mesmo.

Este método consiste na observação de usuários em um ambiente controlado, interagindo com o *software*. Estes testes foram realizados com alguns monitores de robótica e foram feitos de forma individual. Os monitores de robótica foram acompanhados por um auxiliar do teste e tentaram realizar um número de tarefas pré-determinadas no *software*. Na mesma sala foi observado via vídeo e anotado as principais ações e observações dos monitores de robótica. Ao final, estas informações foram compiladas em um relatório de recomendações que está descrito no Anexo G.

Os testes realizados foram: os Testes de Exploração, Testes de Avaliação e os Testes de Validação. Para o presente trabalho são mostrados os resultados dos primeiros testes e a partir dos seus resultados propomos melhorias para uma versão futura do *software*.

A vantagem deste método de avaliação é que pode ser utilizado durante todo o ciclo de desenvolvimento e manutenção do projeto. O destaque deste método é realizar a validação com o usuário final que é uma das estratégias mais sólidas para que se garanta um resultado efetivo na navegação do usuário do *software* e, por consequência, sua satisfação.

O *software* ao qual estamos realizando esses testes vem sendo desenvolvido de forma continuada e participativa, tendo todas as suas ferramentas idealizadas, projetadas e depuradas segundo as necessidades relatadas pela equipe de robótica educacional do laboratório NatalNet.

A sua primeira versão foi desenvolvida em 2006 por [Santos et al. 2006]. Por se tratar de um *software* com requisitos baseados em pesquisas e experimentos, o modelo de engenharia de *software* que está sendo seguido é o modelo Evolucionário [Sommerville 2003].

Este modelo descreve um processo no qual o *software* deve ser desenvolvido de forma a evoluir a partir de protótipos iniciais. Este modelo é baseado na prototipação (ou prototipagem), que vem a ser uma abordagem baseada numa visão evolutiva do desenvolvimento de *software*, afetando o processo como um todo.

Esta abordagem envolve a produção de versões iniciais - protótipos - de um *software* futuro com o qual se podem realizar verificações e experimentações para se avaliar algumas de suas qualidades antes que o *software* venha a realmente ser construído. Um dos principais motivos da escolha deste modelo de engenharia de *software* foi a possibilidade dessas versões iniciais subsidiarem questões inicias do *software* que, uma vez respondidas essas questões, pudéssemos desenvolver novos requisitos a respeito do *software* futuro.

Por estar direcionada a criação de protótipos, não significa que não se tenha uma documentação formal dos requisitos do *software* deste modelo. A partir da versão 1.1, foi desenvolvido um documento de caso de uso com os requisitos que foram pesquisados. Estas informações são encontradas no Capítulo 4, na parte de implementações.

O *software* **RoboEduc** foi estruturado também considerando conceitos pedagógicos, uma vez que se trata de um *software* com finalidades educacionais. Como colocado anteriormente, o *software* apresenta módulos de programação com níveis de complexidade distintos, o que exigirá a cada etapa um grau maior de abstração por parte do usuário-aluno.

Neste sentido, buscamos inicialmente desenvolver conceitos de programação e interface gráfica considerando o contexto de conhecimentos prévios dos alunos, no sentido de levar para o *software* **RoboEduc** ícones, palavras (português) e símbolos, que pressupomos que já estejam internalizados pelos mesmos, tendo em vista o contato desses usuários com o universo da informática (computadores). Universo esse (Re) explorado pelo professor de robótica educacional anteriormente a apresentação do *software*.

Corroboramos com o pensamento piagetiano quando acreditamos que o processo de evolução cognitiva do aluno ocorre por meio de equilibrações (assimilação + acomodação do conhecimento) sucessivas, o que traz como consequência ações e pensamentos mais elaborados, o que acarreta uma determinada maturação cognitiva.

1.4 Organização

A presente dissertação se desenvolve em 5 capítulos, organizados conforme descrito a seguir.

No Capítulo 2 é apresentado o estado da arte ao qual esse trabalho se enquadra.

Apresentamos seus aspectos teóricos com um resumo de alguns temas relacionados a linguagem de programação e alguns trabalhos relacionados, com uma tabela comparativa sobre os principais aspectos que o *software* voltado para robótica educacional precisa. Além de uma tabela comparativa sobre as versões do *software* **RoboEduc**.

No Capítulo 3, é apresentada a descrição do presente trabalho fazendo um breve histórico do desenvolvimento do *software*. Em seguida, é realizada a descrição da definição do problema e, logo após,é apresentada a proposta de solução com os aspectos da modelagem do *software*. É exposta tanto a modelagem utilizada quanto a forma da análise dos resultados dos testes de usabilidade. Abordamos também o modelo proposto para o teste de validação do *software* **RoboEduc**.

No Capítulo 4, mostram-se as implementações iniciais do *software* e suas versões subsequentes e como foi feita a sua evolução baseada nos resultados dos testes de usabilidade, que permitiram criar novos requisitos para versões futuras.

Por fim, no Capítulo 5, as conclusões gerais são abordadas focando os pontos positivos e as dificuldades existentes para o desenvolvimento do plano de validação do *software*. Encontramos também algumas publicações que corroboram o trabalhado desenvolvido.

Capítulo 2

Estado da Arte

Com a difusão da robótica educacional, várias pesquisas têm sido realizadas com o intuito de produzir material para que se possa utilizar esta ferramenta para a educação digital.Segundo Vallejo [Vallejo & Zwierewicz 2007], a educação digital configura novas formas de aprender e ampliar possibilidades para o desenvolvimento de uma perspectiva educativa mundial de aprendizagem sem limitações de espaço e tempo. Ou seja, sem exigir a presença física de todos os docentes e discentes em um espaço geograficamente comum, além de inovar o tempo para a aprendizagem.

Nesta direção, este trabalho vem a contribuir com esta evolução. Neste capítulo é descrito um breve embasamento teórico sobre o conceito de linguagem de programação, para uma melhor compreensão do leitor, como também é feita uma análise sobre os *softwares* disponíveis para tal propósito, como o Scratch [Malan & Leitner 2007], Alice [Wang et al. 2009], RoboLab [Wang & Wang 2001], Imagine [de Abreu et al. 2002], SuperLogo [Chella 2002] e Legal [de Educação Tecnológica da PETe 2008].

2.1 Embasamento teórico

Segundo o dicionário Aurélio [Aurélio 2010], o termo **linguagem** é:

" *O uso da palavra articulada ou escrita como meio de expressão e de comunicação entre pessoas ou a forma de expressão pela linguagem própria de um indivíduo, grupo, classe, etc.*"

Ou seja, a linguagem designa um sistema organizado de símbolos, complexos, extensos e com propriedades particulares. Ela desempenham uma função de codificação, estruturação e consolidação dos dados sensoriais, transmitindo-lhes um determinado sentido ou significado e permitindo ao aluno comunicar as suas experiências e transmitir os seus saberes.

Diante desta definição, analisamos que o termo **linguagem de programação**, segundo o Aurélio, é:

Conjunto de instruções e regras de composição e encadeamento, por meio do qual se expressam ações executáveis por um computador, seja diretamente, seja por meio de processos de compilação, interpretação ou montagem.

Avaliamos que uma linguagem de programação permite ao programador especificar precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias. O conjunto de palavras (*tokens*), compostos de acordo com essas regras, constituem o código fonte de um *software*. Esse código fonte é depois traduzido para código de máquina, que é executado pelo processador.

As linguagens de programação podem ser usadas para criar programas que controlam o comportamento de uma máquina, para expressar algoritmos com precisão, ou como um modo de comunicação humana.

Muitas linguagens de programação têm algum tipo de especificação por escrito da sua sintaxe (forma) e semântica (significado). Algumas linguagens são definidas por um documento de especificação. Por exemplo, a linguagem de programação C é definida por uma norma ISO.

A maioria das linguagens de programação descreve um estilo imperativo, ou seja, como uma sequência de comandos, apesar de existirem algumas linguagens, tais como aquelas que suportam programação funcional ou lógica de programação, que utilizam formas alternativas de descrição.

Existem dois modelos de linguagem de programação convertida e traduzida em código de máquina. Este mecanismo é feito através da compilação e interpretação respectivamente. Este processo é chamado de tradução.

Caso o método utilizado traduza todo o texto do programa (também chamando de código fonte), para depois executar o programa, então se diz que o programa foi compilado e que o mecanismo utilizado para a tradução é um compilador. A versão compilada do programa é armazenada, de forma que o programa possa ser executado um número indefinido de vezes sem que seja necessária uma nova compilação.

Uma linguagem interpretada não é compilada, ao invés disso, cada linha é processada pelo interpretador que a executa. Esse processo é relativamente lento, mas possui a vantagem de não exigir compilação completa para cada mudança feita no seu código.

2.2 *Softwares* educacionais

Neste trabalho observamos uma certa escassez de kits de robótica educacional e, principalmente, de *software* com linguagens de manipulação adequadas ao nosso público. Em geral, os kits existentes no mercado tem um custo elevado, como o kit LEGO MINDSTORMS [Group 2010] e o kit Vex [Vex 2010]. Alguns possuem um preço mais acessível, como o Elenco Beetle [Beetle 2010] e o brasileiro Curumim [Curumim 2010], entretanto, estes permitem somente criar um formato limitado de robôs. Dentre esses kits, o da Lego é o que possui as peças mais interessantes e adequadas para crianças, feitas com material plástico colorido, enquanto a maioria dos outros kits é composto por peças metálicas. Mas, ainda assim, algumas das peças do kit Lego são muito pequenas ou complexas de se encaixar para as crianças abaixo dos 7 anos. Existe também o kit Alfa Hobby da PNCA [PNCA 2010], um kit brasileiro referenciado no Guia das Tecnologias do MEC do ano de 2009 [da Educação do Brasil n.d.], porém ele é direcionado ao público a partir do ensino fundamental.

Nesta seção são descritos alguns *softwares* educacionais existentes no mercado. Foi feita uma análise descrevendo as vantagens e desvantagens de cada um deles, assim como o público ao qual eles são destinadas e qual o seu foco de aprendizagem.

2.2.1 Scratch

O Scratch é um "ambiente de programação", desenvolvido pelo Laboratório Media do *Massachussetts Institute of Technology (MIT)* que está voltado à criação de animações, jogos e arte interativa. Não é um ambiente especifico para se trabalhar com robótica educacional. Segundo [Malan & Leitner 2007], o Scratch destina-se a reforçar o desenvolvimento da fluência tecnológica (entre os jovens) em comunidades economicamente desfavorecidas. Isso poderia contribuir no processo de letramento digital, mas de forma restrita, em virtude do ambiente ser propício apenas para o ensino de linguagens de programação. Como o foco do nosso projeto [de Informática-UFRN 2010] é o letramento digital por meio de uma ambiente de robótica educacional vimos neste ambiente apenas uma das fases do nosso projeto que é a programação. Assim, a programação em si não é nosso foco. Este Ambiente é ilustrado na Figura 2.1

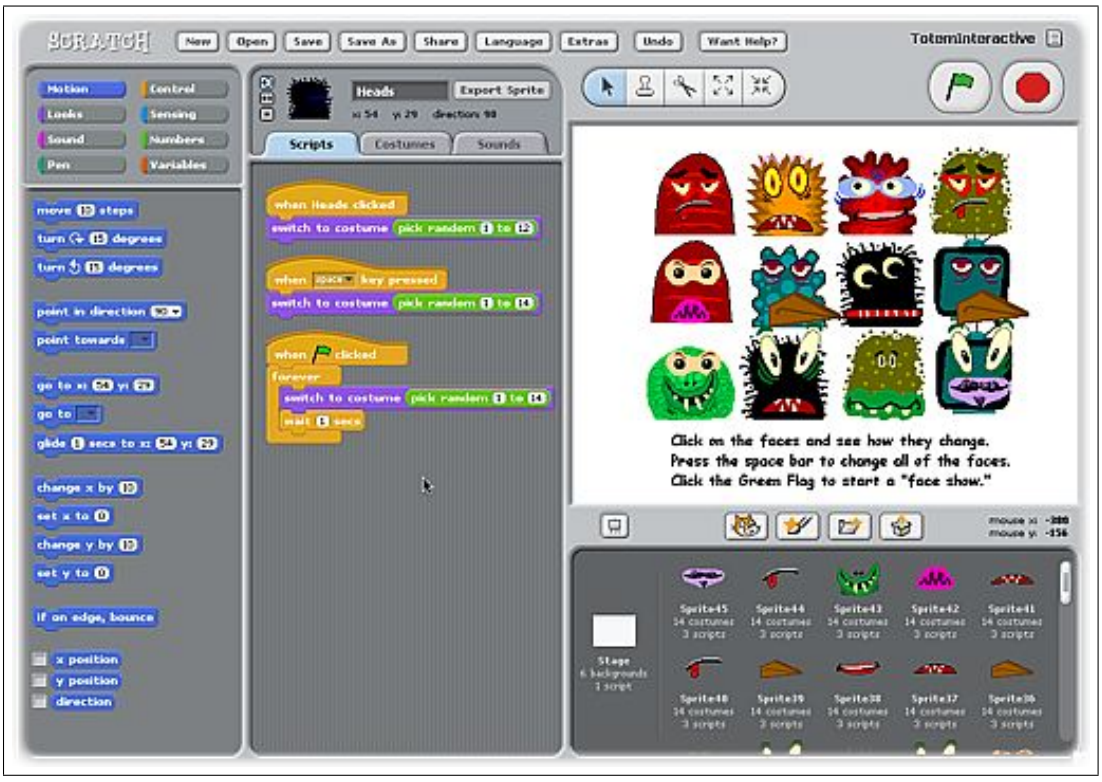

Figura 2.1: Tela principal do Scratch

2.2.2 Alice

Alice [Wang et al. 2009] é uma linguagem de programação destinada a alunos do ensino médio, com o objetivo de ensinar conceitos de programação, como: variáveis, expressões aritméticas, estruturas de condição, estruturas de repetição e funções. Não é uma linguagem voltada para o ensino da robótica educacional. Os resultados das pesquisas realizadas com esta linguagem não mostraram diferenças significativas para a motivação da aprendizagem de programação de computadores. O trabalho [Wang et al. 2009] expõe ainda que a linguagem Alice não mostrou nenhum diferencial com relação ás outras linguagens tradicionais como C++ ou Java. Inferimos, portanto, que a presente linguagem não se adéqua satisfatoriamente ao ensino da robótica educacional. Este ambiente está ilustrado da Figura 2.2

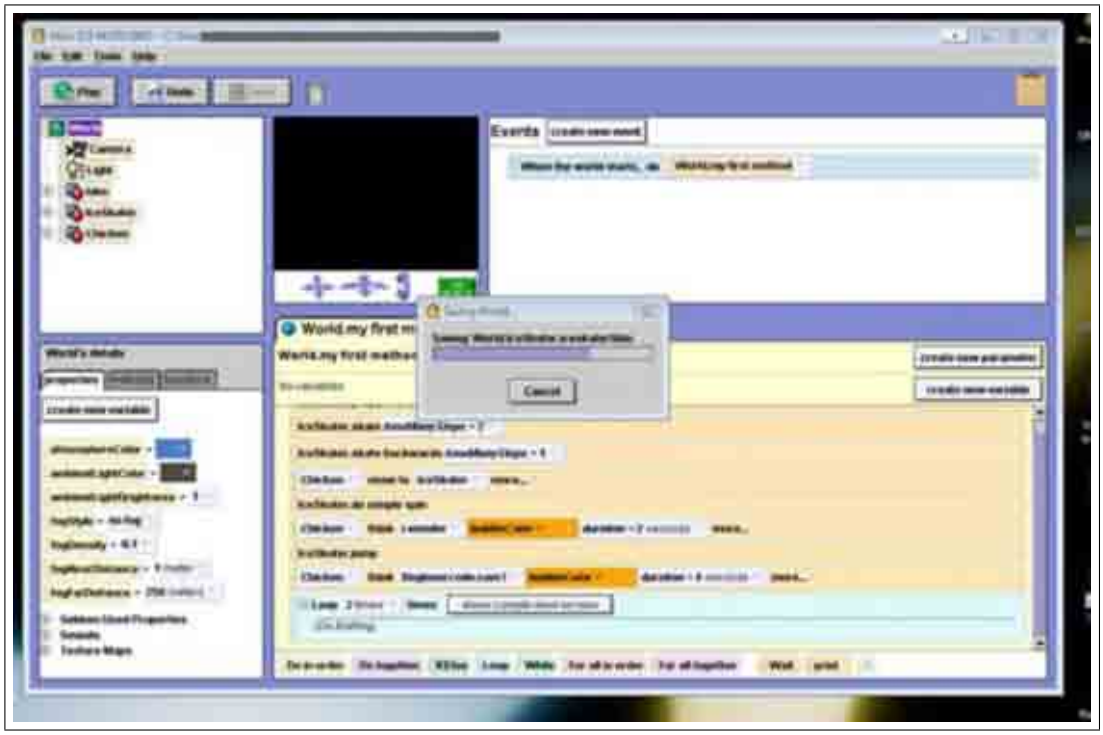

Figura 2.2: Tela principal do Alice

2.2.3 RoboLab

O RoboLab [Wang & Wang 2001] é um ambiente de programação que vem junto aos kits da linha Lego *Mindstorms* [Group 2010], atualmente na versão 2.5. O *software* RoboLab é dividido em três níveis de programação e permite que o nível de programação combine o conhecimento e as habilidades do estudante, como o Piloto, Inventor e Investigador. Em todas essas etapas, a programação é realizada através de ícones que tem por base a linguagem de programação gráfica LabVIEW [Corporation 2010]. Este *software* requer certo conhecimento e compreensão das funcionalidades de baixo nível do robô (motores, sensores, etc.). É ideal para projetos de nível médio ou elementar. Uma parte deste ambiente está ilustrada na Figura 2.3

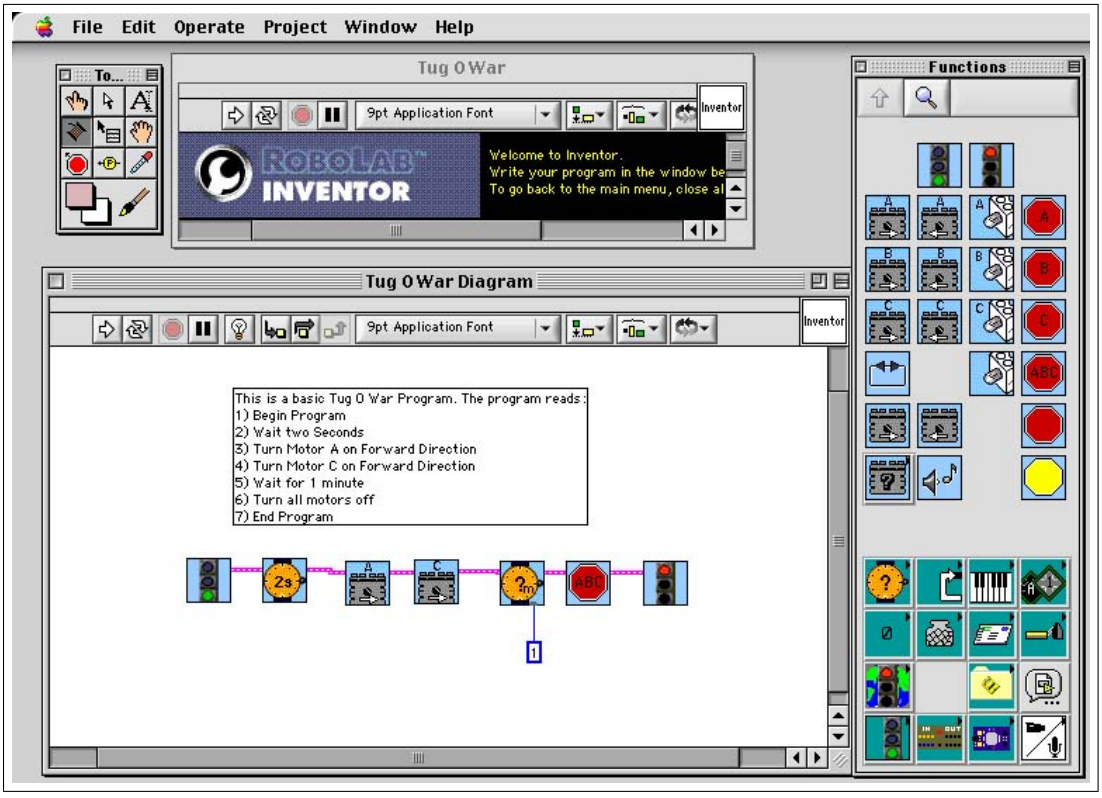

Figura 2.3: Tela principal do RoboLab

2.2.4 Imagine

O Imagine [de Abreu et al. 2002] é um *software* de autoria em linguagem de programação LOGO [Foundation 2010], para ser usado por alunos desde a educação infantil até desenvolvedores profissionais de *softwares* educacionais. O Imagine possui ferramentas para produção gráfica - LogoMotion. Dentre as suas muitas funcionalidades podemos destacar:

- Criação de animações;
- Produção de material para *Web*;
- Criação de ambientes multimídia;
- Possui entrada e saída de voz;
- Desenvolve ambientes com modelagem;
- Constrói utilizando o método arrastar e soltar objetos;
- Comunica ideias;
- Constrói apresentações;
- Permite desenvolver projetos colaborativos;
- Constrói programas;
- Trabalha com controle e manipulação de dados;
- Compõe músicas.

Imagine utiliza o recurso da linguagem Logo com o paradigma de programação orientado a objeto. Imagine permite ainda enviar dados, objetos e instruções via rede. Imagine tem saída para periféricos que permitem trabalhar com robótica.

Este *software* possui muitos recursos disponíveis, contudo essa variabilidade de comandos presentes pode acarretar uma perda de foco na aprendizagem do usuário, uma vez que os recursos não são introduzidos por níveis de aprendizagem. Não apresenta uma interface lúdica, amigável, motivadora. Este ambiente é ilustrado na Figura 2.4

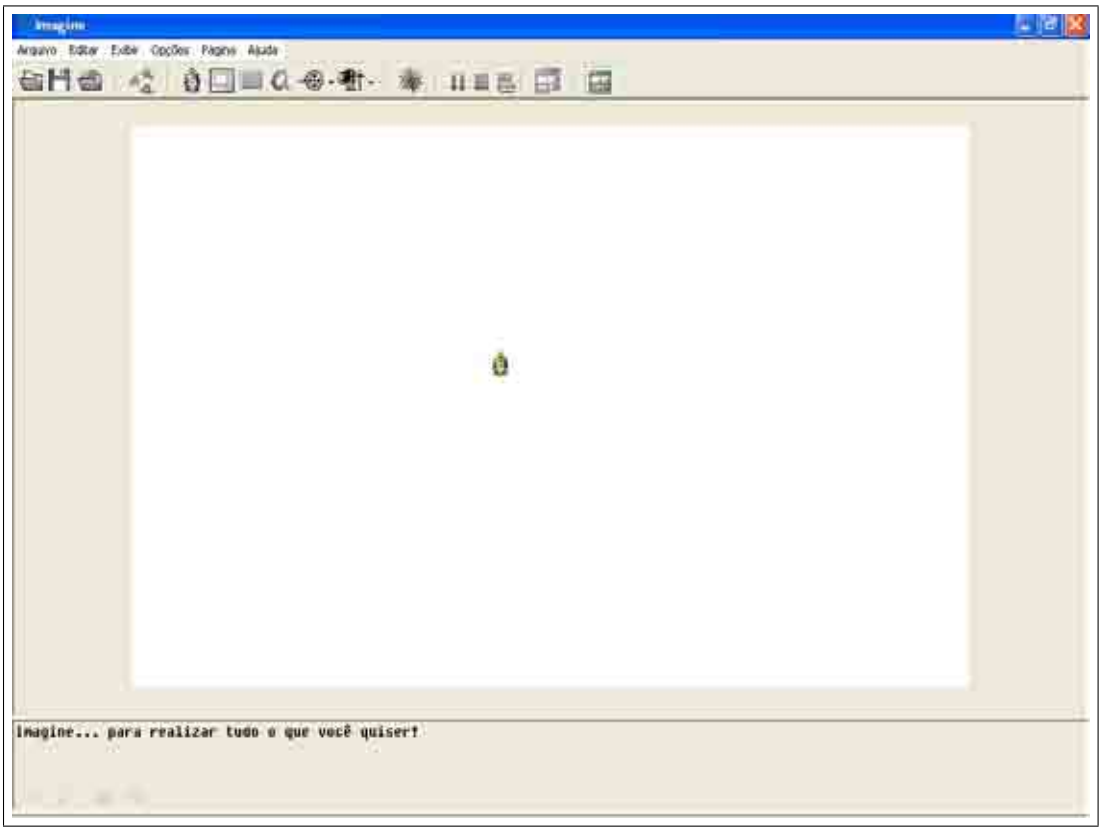

Figura 2.4: Tela principal do Imagine

2.2.5 Super Logo

SuperLogo [Chella 2002] é uma versão do LOGO [Foundation 2010] desenvolvida, por pesquisadores do *Massachussetts Institute of Technology (MIT)*. Essa linguagem apresenta um grande grau de flexibilidade, podendo ser utilizada tanto por crianças como por programadores experientes, atendendo, em ambos os casos, as necessidades do usuário. Tem-se uma terminologia simplificada, pela facilidade no que diz respeito a termos de nomes de comandos, de regras sintáticas e de uma parte gráfica. Um ponto bastante interessante no Logo e, consequentemente, no SuperLogo é a parte gráfica caracterizada pela presença de um cursor representado pela figura de uma tartaruga que pode ser deslocada no espaço da tela através de alguns comandos relacionados ao deslocamento e giro da mesma. Porém, esta linguagem não fornece suporte ao envio de comandos para um robô construído fisicamente. Este ambiente está ilustrado na Figura 2.5

Figura 2.5: Tela principal do SuperLogo

2.2.6 Legal

Por fim, a linguagem LEGAL [de Educação Tecnológica da PETe 2008] está inserida dentro do Ambiente de Programação LEGAL que vem junto ao kit de robótica educacional da PNCA. O seu objetivo é iniciar o trabalho na robótica e na mecatrônica. Este *software* permite ao usuário projetar, construir e programar robôs e dispositivos mecatrônicos. Todas as ações que o robô deve executar serão definidas pelo usuário no ambiente de programação LEGAL. Os programas são escritos utilizando a linguagem LEGAL. O programa, uma vez escrito e compilado, será descarregado no módulo de controle do robô via cabo serial. Ao término deste processo, o robô estará pronto para funcionar de forma autônoma e poderá ser desconectado do computador principal. Porém, esta linguagem de programação é limitada ao número de sensores, pois tem uma programação orientada a eventos. Um sinal capturado por um sensor dispara um desses eventos, porém não é possível programar uma função que interaja com vários sensores ao mesmo tempo. Este ambiente de programação esta ilustrado na Figura 2.6

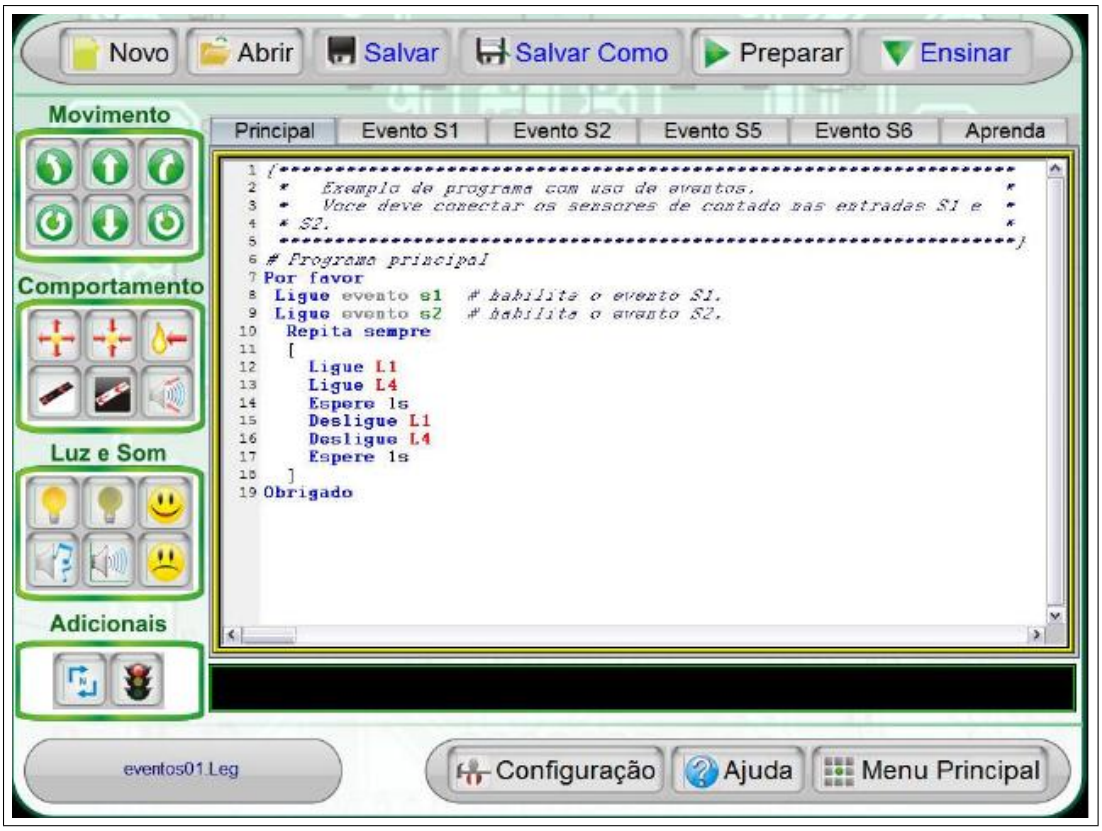

Figura 2.6: Tela principal do Legal

2.2.7 Análise dos trabalhos

Existem outros *softwares*, além dos citados anteriormente, livres e comerciais, para programação de robôs. Por exemplo, quando falamos de linguagens de programação para o kit da LEGO *MINDSTORMS* [Group 2010], as linguagem como *brickOS* [brickOS 2010]e NQC [NQC 2010] são baseadas na linguagem de programação C e não são simples para as crianças entenderem. O mesmo acontece com o programa *leJOS* [LeJOS 2010] baseado na linguagem de programação Java.

Após listar esses *softwares* e conhecer suas características e finalidades constatamos que são *softwares* para serem utilizadas com pessoas que tenham um embasamento teórico sobre robótica, programação e informática. Conforme pode ser visto na Tabela 2.1. Porém, essa não é realidade da sociedade atual como foi explicado no Capítulo 1. Assim, esses *softwares* não são adequados para serem utilizados como um instrumento para educação digital. Suas interfaces, embora sejam através de ícones, são bastante difíceis de serem compreendidas, fazendo com que os usuários não utilizem todos os recursos disponíveis no *software*.

Cada ponto analisado na tabela foi escolhido baseando-se nos conceitos de programação, robótica e letramento que o usuário precisa ter conhecimento prévio. Como critério de preenchimento da tabela foram feitos testes de usabilidade nos *software* citados e colocamos "sim"quando era necessário utilizar o conceito para realizar alguma tarefa proposta pelo teste e "não"caso contrário. Foi um preenchimento empírico.

	Scratch	**Alice**	**Imagine**	**RoboLab**	**SuperLogo**
Linguagem Base	Squeak	C	Logo	LabView	Logo
Entrada de Dados	texto	texto	texto	visual	texto
Necessário Conhecer:					
Programação	Sim	Sim	Sim	Não	Sim
Robótica	Sim	Sim	Sim	Sim	Não
Escrita	Sim	Sim	Sim	Não	Sim
Utiliza Conceitos de:					
Motores	Sim	Sim	Sim	Sim	Não
Sensores	Sim	Sim	Sim	Sim	Não
Programação Básica	Sim	Sim	Sim	Não	Sim
Controle de Fluxo	Sim	Sim	Sim	Sim	Sim
Programação Avançada	Sim	Sim	Sim	Não	Não

Tabela 2.1: Análise comparativa dos *softwares* de robótica educacional

A seguir é apresentada a Tabela 2.2 com a evolução dos níveis de programação da linguagem **Educ**.

Níves do *Software*	**Controlar**	**Ensinar**	**Nível 1**	**Nível 2**	**Nível 3**	**Nível 4**	**Nível 5**
Linguagem Base	NXC	NXC	NXC	NXC	NXC	NXC	NXC
Entrada de Dados	visual	visual	visual	visual	texto	visual	texto
Necessário Conhecer:							
Programação	Não	Não	Não	Sim	Sim	Sim	Sim
Robótica	Não	Não	Não	Não	Não	Sim	Sim
Escrita	Não	Não	Não	Não	Sim	Não	Sim
Utiliza Conceitos de:							
Motores	Não	Não	Não	Não	Não	Sim	Sim
Sensores	Não	Não	Não	Não	Não	Sim	Sim
Programação Básica	Não	Não	Sim	Sim	Sim	Sim	Sim
Controle de Fluxo	Não	Não	Não	Sim	Sim	Sim	Sim
Programação Avançada	Não	Não	Não	Não	Não	Sim	Sim

Tabela 2.2: Análise evolutiva dos níveis do *software* **RoboEduc**

Capítulo 3

Avaliação e Validação do RoboEduc

O presente trabalho é a avaliação e validação do desenvolvimento do *software* **RoboEduc**. Desde o início do seu desenvolvimento foi adotado o modelo evolucionário de engenharia de *software*, como foi descrito na metodologia no Capítulo 1. Este trabalho além de avaliar as últimas versões do *software* também é uma evolução dos protótipos anteriores.

Neste capítulo levantamos alguns questionamentos sobre o tema da nossa pesquisa.

3.1 Definição do problema

Conforme descrito no Capítulo 1, avaliando o quadro discente atual da nossa pesquisa, observados um público heterogêneo, porém, ainda com um número significativo de analfabetos digitais no contexto de um processo de educação digital.

Com base nas pesquisas foram levantados alguns questionamentos:

1. Os kits existentes conseguem incluir pessoas pertencentes aos três grupos da sociedade: migrantes, nativos e analfabetos digitais?
2. Por que os kits atuais não conseguem atacar esse público?
3. Quais os melhores kits para isso?
4. Que mudanças eles precisam sofrer para melhorar sua usabilidade?

Para responder a essas questões, temos como base os estudos realizados e descritos no Capítulo 2, onde constatamos que são *softwares* para serem utilizadas com pessoas que tenham um embasamento teórico sobre robótica, programação e informática. Logo, se são necessários conhecimentos prévios, os analfabetos digitais não seriam contemplados no processo de inclusão digital.

Os kits atuais não conseguem atacar esse público, pois trazem muitas informações ao mesmo tempo, tornando o processo de aprendizado confuso. Uma vez que a robótica educacional está como um auxiliador no processo de inclusão digital e não com objeto principal de aprendizagem.

Dentre as mudanças necessárias para atingir uma sociedade tão heterogênea, está a divisão dos conhecimentos por categorias. Assim propomos uma linguagem de programação multiplataforma e representada em níveis diferentes de abstração. Esta linguagem

está associada a uma metodologia de ensino, em um *software* composto por um ambiente de tutoria para a montagem de robôs para diferentes fins: Ambiente de controle remoto de robôs e ambientes de programação em diferentes abstrações gráficas e textuais.

O desafio deste trabalho é fornecer ao docente um auxílio na educação digital contemplando as três níveis da sociedade atual ao mesmo tempo. Assim, será possível que os discentes, com diferentes níveis de cognição, possam realizar a mesma atividade proposta pelo docente respeitando as suas limitações e contemplando as suas competências.

3.2 Solução proposta

Diante dos problemas levantados temos como objetivo responder o seguinte questionamento: Existe o problema de incluir alunos de todos os tipos. Como fazer isso?

Embasado nos resultados das pesquisas do grupo desde 2006, vimos que a robótica educacional se apresenta com o objetivo de tornar o aprendizado mais significativo por mobilizar, através de seu uso pedagógico, diferentes tipos de conhecimento e competências. [da Silva et al. 2008].

Uma das ferramentas utilizadas neste processo são os kits de robótica que são implementações, dentro da robótica educacional, onde os alunos terão a oportunidade de adquirir meios de solucionar problemas constante. Além disso, o professor poderá demonstrar na prática muitos conceitos teóricos, até os de difícil compreensão, motivando todos os envolvidos na atividade.

Como as implementações atuais de kits (lego, legal, etc) não se adéquam ao nosso público alvo, conforme mostram as Tabelas 2.1 e 2.2, onde todos foram comparados, observou-se que os kits realmente não atendiam nossos requisitos de público.

Foram realizados testes de usabilidade e observou-se que o **RoboEduc** apresentou um melhor desempenho para esse caso específico de público heterogêneo, mas mesmo assim ainda existem alguns pontos a serem melhorados.

O *software* **RoboEduc** e a linguagem **Educ** foram idealizadas pelo educador Luiz Marcos Garcia Gonçalves, professor na área da computação, na Universidade Federal do Rio Grande do Norte, Brasil.

Este *software* é utilizado para trabalhar com crianças e adolescentes, desde dos seus primeiros projetos. Desde sua criação em 2004 até 2006, o **RoboEduc** ficou restrito a estudos e aplicações do laboratório NatalNet da Universidade Federal do Rio Grande do Norte.

A preocupação inicial do grupo de pesquisa concentrava-se principalmente em desenvolver um *software* educacional. Posteriormente, o foco passou para o desenvolvimento de um *hardware* para implementar o interpretador/compilador para a linguagem **Educ** e demonstrar seu poder de desempenho, tendo em vista principalmente sua relação com a robótica no ensino infantil.

Uma das vantagens propostas para esta linguagem é que a mesma funciona como um intermediador entre as diversas linguagens de baixo nível. A linguagem **Educ** é configurável a qualquer linguagem de baixo nível para controle de dispositivos robóticos.

O *software* **RoboEduc** nasceu com base nas referências teóricas de aprendizado de [da Silva 2009]. A partir do projeto de Inclusão Digital com Robôs [RoboEduc 2010],

iniciado em 2006 na Escola Pública Professor Acendido de Almeida, Natal -RN. Foi utilizado o *software* **RoboEduc** em microcomputadores e kits Lego *Mindstorms* aplicados em 20 alunos do quarto e quinto ano, pode-se dizer que o *software* **RoboEduc** iniciou a sua saída dos laboratórios e penetrou na escola.

O *software* **RoboEduc** tem uma linguagem de programação, onde ocorre a interação entre os robôs e os usuários. A principal diferença entre a linguagem **Educ** e as outras linguagens de programação está no alcance do seu público alvo. Esta linguagem foi desenvolvida para atender, além do público adulto e juvenil, alunos inseridos na educação infantil, ou seja pode ser usada por crianças. Esta linguagem possibilita outros aprendizados como: Relação ícone - palavra - ação, lateralidade, coordenação motora, raciocínio lógico, dentre outras competências.

A linguagem **Educ** está embutida em uma metodologia educacional que respeita o nível cognitivo do aluno, onde a criança aprende explorando o seu ambiente, com regras que ela mesma impõe.

A linguagem **Educ** é uma linguagem simples e abrangente. Simples, porque é fácil de aprender: pessoas em processo de alfabetização, de qualquer idade, podem programar em seu primeiro contato com a linguagem. Ao contrário de outras linguagens, esta permite a pessoa programar sem a necessidade de muitos conhecimentos prévios. É abrangente, porque terá recursos sofisticados que atendem às exigências de todos os inseridos nos mais variados tipos de gradação de acesso à tecnologia.

Apesar de ser uma linguagem acessível aos migrantes e aos analfabetos digitais, a **Educ** não é uma linguagem que atende somente estes tipos. Através dela, os nativos digitais aprendem explorando, investigando e descobrindo por si ou com o direcionamento de um monitor. Inclusive, é possível trabalhar com a linguagem **Educ** e robótica educacional com crianças a partir do ensino infantil (quatro anos).

Ensinar a linguagem **Educ** por si só não é a questão, nem resolverá os problemas do letramento digital. É necessário pensar na abordagem e na sua metodologia. Tentar "ensinar" linguagem **Educ** talvez seja uma boa iniciativa. Por outro lado, é possível considerar a utilização da linguagem **Educ** fora do contexto escolar. Devemos considerar o seu uso em outros ambientes: em casa, em projetos pessoais, etc. O ser humano está sempre aprendendo e a escola não é o único lugar em que isso pode ocorrer.

A linguagem **Educ** aliada a um robô permite fazer coisas que as outras linguagens tradicionais não conseguem: o retorno imediato. Através de experiências em escolas públicas ministrando oficinas de robótica, entendemos que o retorno imediato é que torna essa linguagem divertida e mais fácil de aprender.

A preferência por diferentes níveis de programação gradativamente mais complexos favorece o amadurecimento mental do aluno, no que diz respeito ao raciocínio lógico exigido ao programar. Os usuários da linguagem **Educ** partem de um sistema simples de programação para um mais complexo, obedecendo níveis de amadurecimento cognitivo conquistados com a prática, recorrentes do uso do *software* **RoboEduc**. Vale ressaltar que não estamos limitando a linguagem apresentada ao uso de crianças. Jovens e adultos poderão fazer uso da linguagem **Educ**, de forma similar ao uso por crianças.

Introduzimos o usuário em um universo de programação inicial (Módulos: controlar e ensinar), que facilmente pode ser aprendido pelo adulto, o que resulta em um tempo a

menos de uso comparado a uma criança, que porventura poderá passar uma maior parte do tempo aprendendo a programação com tais módulos. Entendemos que o avanço na aprendizagem depende do nível de conhecimento do usuário, o que ele já carrega com sua vivência com o meio e que adquire manipulando o nosso *software*.

Os níveis seguintes (programação gráfica e textual) exigem mais intimidade com a programação, aumentando a complexidade do que é exigido do usuário. Estes níveis, a nosso ver, são extensões dos primeiros, os quais seguem uma sequência avançada de abstração. Entendemos que se o aluno passa pelos primeiros níveis da linguagem **Educ**, se tornará mais fácil utilizar os níveis de programação seguintes. A linguagem **Educ** dá ênfase a familiarização do aluno-usuário com o conceito e práticas de programação.

O usuário é estimulado a usar o ambiente **RoboEduc**, este amigável, lúdico e instigante ao uso, para responder inquietações provocadas pelo ambiente desafiador da robótica educacional. Desta forma, a linguagem **Educ** motiva o aluno a usar seus níveis, do mais simples ao mais complexo, para se ambientar com o *software* e aprender conceitos que envolvem a prática da programação de robôs.

A especificação da linguagem **Educ** juntamente com a sua gramática são explicados no trabalho [Barros 2008].

Para validarmos a solução proposta foram realizados testes de usabilidade. Na Seção 3.3 são descrito todos os critérios que foram utilizados assim como a forma que foram realizados.

3.3 Teste de usabilidade

Segundo [Carvalho 2002], a usabilidade está diretamente relacionada com a interface que, juntamente com o usuário e o sistema computacional interativo, constituem os três principais componentes da Interação Homem-Computador (IHC).

O teste de usabilidade é um processo no qual amostras de usuários avaliam o grau que o *software* se encontra em relação a critérios específicos de usabilidade [Ferreira 2002].

Na literatura, a quantidade de parâmetros para medir usabilidade é variável. Para Nielsen [Nielsen 1993], há cinco parâmetros:

1. Fácil de Aprender;
2. Eficiente para Usar;
3. Fácil de Lembrar;
4. Pouco Sujeito a Erros;
5. Agradável de Usar.

O primeiro item analisa se o usuário consegue rapidamente interagir com o *software*, compreendendo as opções de navegação e as funcionalidades dos botões. O segundo analisa se, após compreender como o *software* funciona, o usuário consegue localizar a informação que precisa. O terceiro item analisa se mesmo um usuário que não utiliza o *software* com uma certa frequência consegue lembrar do que já foi aprendido ou é necessário retornar aos dois itens anteriores. O quarto item analisa se o *software* não induz

o usuário a cometer muitos erros e se esses erros são críticos ao *software*. E o último item a ser analisado refere-se a satisfação dos usuários em interagir com o *software*.

Diante desses parâmetros, o referido teste pode servir para diferentes propósitos. Envolvendo vários tipos de tarefas como: medidas de desempenho, disposição de escalas, entrevistas ou inspeções a serem aplicadas. Buscando com isto encontrar problemas de usabilidade e fazer recomendações no sentido de eliminar os problemas e melhorar a usabilidade do *software*.

Uma forma interessante de determinar os tipos de testes é através do ciclo de vida de desenvolvimento do *software*. Desta forma, se alguma deficiência é perdida em um teste, um outro ciclo de teste oferece a oportunidade para identificar esta deficiência. Os tipos de teste são apresentados logo a seguir e ilustrados na Figura 3.1

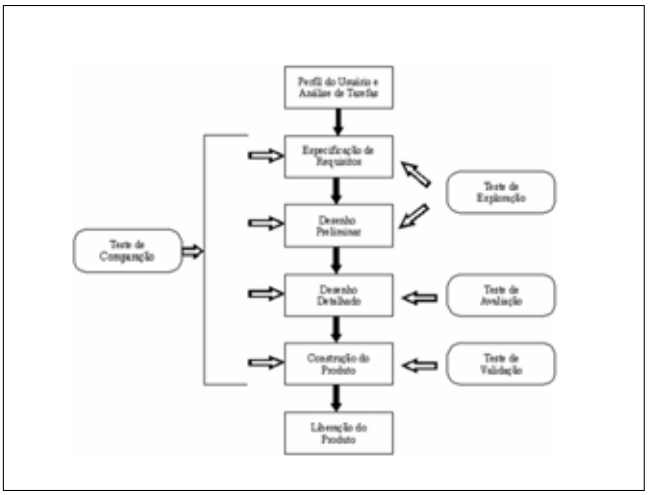

Figura 3.1: Ciclo de vida de desenvolvimento do *software* **RoboEduc**

Seguindo a Figura 3.1, sua primeira etapa é a definição do **Perfil do Usuário e Análise de Tarefas**. Para isso foi elaborado um questionário Anexo D. Este é um item de extrema importância para o sucesso do teste, pois um mesmo *software* pode ser excelente para um determinado grupo de usuários e inadequado para outros. Este questionário busca os aspectos da experiência computacional, o nível educacional, idade, sexo e o estilo de aprendizado.

Os dois itens a seguir estão englobados no Teste de Exploração, que são as **Especificações de Requisitos** e os **Desenhos Preliminares**. Este teste foi realizado quando o *software* estava na fase de definição em 2006. Porém, houveu reformulações ao longo desses anos e sempre que uma nova interface era proposta este teste era realizado com o objetivo de avaliar se o novo desenho proposto era viável ou não para a implementação.

Este processo ocorreu de forma bastante informal, onde a equipe de desenvolvimento realizava encontros semanais para a discussão dos desenhos para a avaliação do *layout* básico, organização dos *menus*, definição das funções e a navegabilidade do *software* como um todo.

Na quarta etapa do clico de desenvolvimento do *software* temos o **Desenho Detalhado**. Nesta etapa foi realizado o teste de avaliação. Foi o mais simples e direto dos

testes. Ele começou quase que em paralelo com as etapas anteriores e perdurou até o meio do ciclo de desenvolvimento do *software*.

O seu propósito foi expandir o que foi conseguido no teste de exploração, avaliando a usabilidade em um nível baixo de operações e aspectos do *software*. Baseando-se no modelo conceitual do *software*, este teste buscou examinar e avaliar como o conceito foi implementado efetivamente, verificando como um usuário consegue desenvolver tarefas reais, identificando deficiências específicas de usabilidade. Este teste foi bastante simples, pois o usuário executou tarefas simples caminhando entre as telas, dando mais ênfase ao comportamento. Medidas quantitativas foram coletadas e modificações foram feitas no *software*. Os primeiros resultados destes testes podem ser visto no Capítulo 4 de implementações, onde é detalhada toda a evolução do *software*, desde sua primeira versão em 2006 até os dias atuais.

As últimas etapas do ciclo de desenvolvimento do *software* são **Construção do Produto** e **Liberação do Produto**, onde contemplam o teste de validação. Este teste nunca foi realizado no *software* **RoboEduc** até a sua versão 3.0. O presente trabalho realizou esta etapa e analisou os resultados obtidos para modificações, o que gerou a versão 4.0 do *software*.

O objetivo deste teste é verificar como o *software* se enquadra em relação a padrões de usabilidade, padrões de performance e padrões históricos. Esses padrões são originados dos objetivos de usabilidade definidos no começo do projeto através de experimentos nas escolas, entrevistas com os alunos-usuários e também pelos estudos-pesquisas da equipe de desenvolvimento. Nós buscamos validar também a interação entre os componentes do *software*, como, por exemplo, a forma como a documentação, a ajuda, o *software* e o *hardware* estão integrados uns com os outros. Neste teste de validação foram enfatizados os julgamentos quantitativos sobre o *software* e um maior rigor experimental. Os materiais necessários para a realização deste teste são:

- Questionário para o perfil do usuário (Anexo D);
- O Plano de teste, que consiste na base de todo o teste, especificando como, quando, onde, quem, o porquê e o quê sobre o teste de usabilidade (Anexo A);
- Lista de tarefas, que incluem as tarefas que serão realizadas pelos usuários durante o teste (Anexo F);
- Roteiro de orientação, que descreve o que irá acontecer durante a sessão de teste, possibilitando uma visão prévia aos participantes (Anexo C);
- Questionário de avaliação do *software* pelo participante, que tem por objetivo colher informações sobre a opinião do participante (Anexo B);
- Tópicos para questionamento, que são tópicos a serem discutidos em uma sessão após a realização do teste do protótipo do *software* (Anexo E).

.

3.3.1 Métodos de avaliação e técnicas para coleta dos dados

Há vários métodos de avaliação de usabilidade, dentre eles podemos citar quatro métodos que a literatura aceita com certa uniformidade: o método da avaliação heurística, o da observação, o da sondagem e o método experimental.

A avaliação heurística é feita por peritos. A observação, que é feita, geralmente, num laboratório, podendo ser direta ou indireta, utilizando-se para o efeito câmaras de vídeo para gravarem a interação do participante com o *software* educacional ou possibilitando ao observador uma situação em que observe sem ser visto.

Pode-se, ainda, solicitar aos utilizadores para verbalizarem o que pensam. Esta verbalização pode ser pedida a um participante, mas é pouco natural, ou ao grupo de participantes enquanto interagem. Ao utilizar o método de observação, é imprescindível que o observador use um guia, assim, saberá o que vai observar e o que se pretende que seja observado em particular. Dá mais trabalho antes da observação, mas depois facilita o recolhimento de dados e a análise dos mesmos.

Na sondagem podem ser utilizadas, basicamente, duas técnicas de coleta de dados: questionários ou entrevistas.

Por fim, temos o método experimental, que permite o controle de variáveis. Preece [Preece et al. 1993] propõe ainda o método analítico ou formal que é aplicado numa fase inicial. Este método permite prever a usabilidade de uma interface antes de esta ser usada.

Para o presente trabalho, os métodos utilizados foram observação e sondagem. Os dados que foram coletados são sobre desempenho e dados preferenciais.

Os dados sobre desempenho representam medidas do comportamento do participante, incluindo erros, número de acessos a ajuda por tarefa, tempo de execução de uma tarefa, dentre outros. Esses dados foram coletados durante a observação do teste.

Os dados preferenciais representam medidas da opinião do participante incluindo respostas a perguntas e posicionamento do participante diante dos demais. Esses dados foram coletados por escrito através de questionário e através de questionamento do participante após o teste.

3.3.2 Análise dos dados

A análise dos dados foi formada por dois processos distintos. O primeiro processo constituiu de uma análise preliminar. O objetivo foi averiguar rapidamente os piores problemas encontrados, de forma que a equipe já pudesse trabalhar neles imediatamente, sem ser necessário esperar pelo relatório final do teste. A análise preliminar foi realizada logo após o teste ter sido completado. Foi elaborado um pequeno relatório escrito.

O segundo processo constituiu de uma análise abrangente, cuja duração levou três semanas após o teste. Este relatório final foi de modo mais exaustivo. O relatório final incluiu todas as descobertas do relatório preliminar (desenvolvido na análise preliminar), devidamente atualizado, além de outras análises e novas descobertas.

Para a elaboração deste relatório foram seguidas algumas recomendações, que segundo [Jeffrey 1994], devem conter para a análise dos dados. Estas recomendações são:

- **Organização dos dados:** Organizar envolve compilar os dados coletados de acordo com padrões pré-definidos. Este mecanismo acelera o processo de análise, verifica se foram coletados os dados corretos, verifica os problemas declarados no plano de teste e ajuda a detectar se algo importante foi esquecido.
  Ao final de cada dia, foram reunidos todos os dados coletados e verificado se estavam legíveis e se os observadores estavam contribuindo efetivamente para a coleta

dos dados. Diariamente, foram transferidas algumas notas para o computador para que fosse mantido um resumo dos dados, como, por exemplo, uma média do tempo gasto nas tarefas.

- **Criar Resumos:** Depois das sessões terem sido completadas, a compilação dos dados foram finalizadas e foi gerado um resumo descritivo dos dados quantitativos. Isto foi realizado através da transferência de informações das planilhas que contêm os dados coletados para planilhas de resumo.
- **Resumo de Dados de Desempenho:** Os dados de desempenho foram resumidos em tarefas cronometradas e tarefas de precisão.
  As tarefas cronometradas apresentam quanto tempo os participantes levaram para completar cada tarefa. Esses dados foram representados em forma de tabela. Onde podem ser facilmente analisados alguns pontos:
  - Média do tempo gasto para completar a tarefa, calculado pelo somatório do tempo gasto por todos os participantes que completaram a tarefa.
  - Tempo mediano, é o tempo que é apresentado exatamente no meio (assinalado na tabela) quando todos os tempos gastos para completar a tarefa são listados em ordem crescente.
  - *Range*, é o tempo mais alto e o tempo mais baixo gasto para se completar a tarefa. Esta estatística é bastante relevante no caso de uma diferença grande entre o menor e o maior tempo apresentado e para se tentar detectar o motivo e ações a serem tomadas para se evitar a ocorrência de diferenças tão grandes.
  - Desvio Padrão (S), tal como o *range*, é uma medida de variabilidade dos dados, ou seja, em que grau os dados se diferem uns dos outros. Sua Fórmula 3.2 é apresentada a seguir:

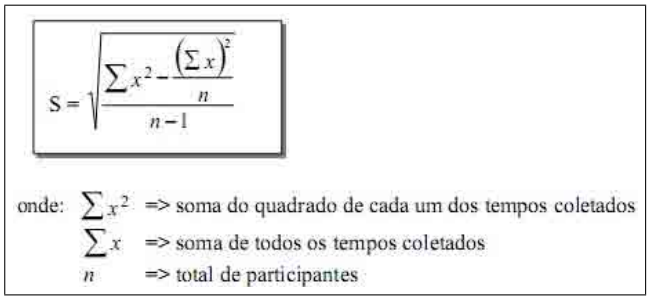

Figura 3.2: Fórmula do desvio padrão (S)

As Tarefas de Precisão englobaram diferentes estatísticas. Foi considerado simplesmente o número de erros ocorridos por tarefa. Foi rastreado o número de participantes que executaram uma determinada tarefa com sucesso e o número de participantes que necessitaram de algum auxílio para executar uma determinada tarefa.

- **Resumo de Dados Preferenciais:** Os dados preferenciais foram originados de várias fontes como, pesquisas e questionários.
  - Questões de escolha: Foi realizada uma contabilização das respostas de cada questão individualmente e foi calculada uma média por item, possibilitando visualizar qual o número de participantes que selecionaram cada resposta.

- Questões abertas e comentários: Foram listados todas as questões e agrupadas as respostas semelhantes em categorias significativas, como, por exemplo, somar todas as referências positivas e negativas sobre um item em particular, possibilitando visualizar os resultados rapidamente através do agrupamento realizado.
- Sessões de questionamento do participante: Foram transcritas todas as entrevistas para o papel, pois a visualização dos comentários escritos facilita a seleção dos comentários críticos.

O relatório completo está no Anexo G.

Capítulo 4

Evolução e Resultados

Como mencionado no Capítulo 1, na Seção 1.3, por adotar o modelo evolucionário de processo de *software*, neste capítulo há um histórico dos protótipos inicias do *software* **RoboEduc**, mostrando os resultados iniciais dos testes de usabilidade e as melhorias que foram feitas até o presente momento.

4.1 Versão 1.0

O **RoboEduc**, em sua primeira versão, apresenta como pré-requisitos de funcionamento um computador (*Desktop*) com a plataforma Linux, uma porta de saída serial e o pacote brickOS [brickOS 2010] instalado. Esta versão apresenta duas funcionalidades: controle remoto e comunicação com o dispositivo robótico. A interface controle remoto da primeira versão do **RoboEduc** é apresentada na Figura 4.1

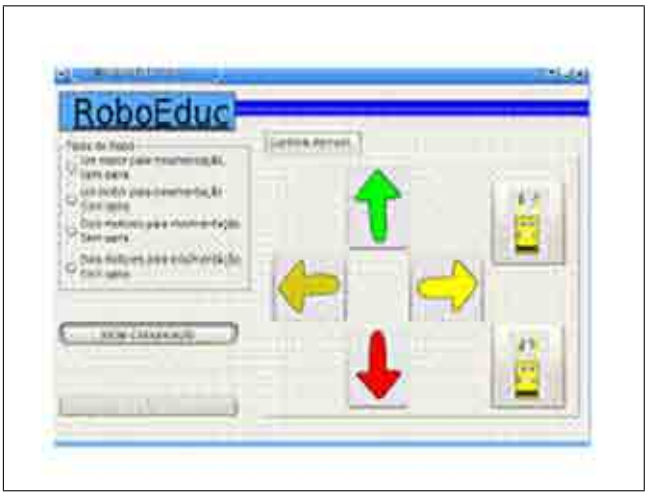

Figura 4.1: Interface da primeira versão

As informações sobre os tipos de robôs que podem ser construídos deverão ser previamente selecionados através do *menu* localizado ao lado esquerdo superior da tela principal

do *software*. Dentre as opções de protótipos que podem ser construídos fisicamente, temos:

1. Um motor para movimentação. Sem garra.
2. Um motor para movimentação. Com garra.
3. Dois motores para movimentação. Sem garra.
4. Dois motores para movimentação. Com garra.

No presente momento, existem quatro tipos de robôs disponíveis para escolha, a diferença entre eles está na quantidade de motores necessários para a locomoção e a presença ou não de uma garra. Na parte central do *software* é exibido todas as ações que o robô pode executar, dependendo de sua construção física, estas ações são: FRENTE, TRÁS, DIREITA, ESQUERDA, PARAR, ABRIR GARRA e FECHAR GARRA.

Após a seleção do robô que foi construído, o *software* habilitará as possíveis ações que podem ser realizadas pelo robô selecionado pelo usuário. Por exemplo: se o usuário selecionou a opção "Um motor para movimentação. Com garra", apenas estariam habilitados os botões das setas indicativas para FRENTE, TRÁS, ABRIR GARRA e FECHAR GARRA. Os demais botões que representam as demais ações seriam desabilitados, uma vez que o robô construído não teria a possibilidade de realizá-las, como pode ser visto na Figura 4.2. Este controle também poderá ser feito por meio do teclado.

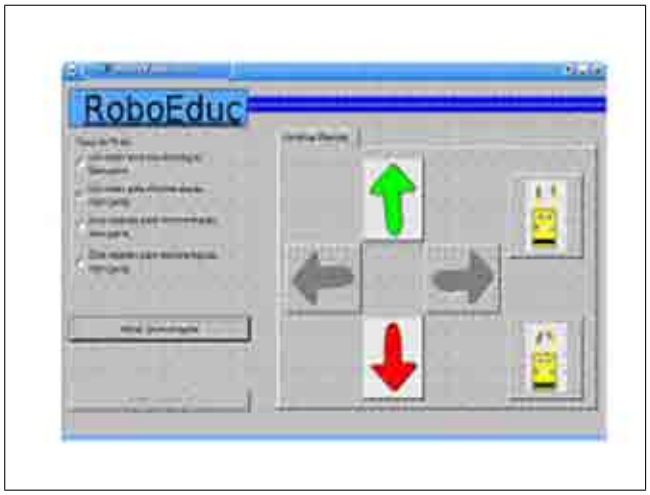

Figura 4.2: Botões habilitados/desabilitados

No lado esquerdo inferior da tela principal encontram-se outros dois botões, o primeiro é o responsável pela inicialização da comunicação entre o computador e o protótipo. O segundo, cuja funcionalidade não foi implementada, iria permitir que o protótipo executasse as mesmas ações realizadas via controle remoto de forma autônoma.

A comunicação necessária entre o *software* e o *Robotic Control eXplorer* (RCX), que é um computador embarcado de propósito específico (uma unidade de controle), é estabelecida através da porta serial do computador onde conectava-se a torre de comunicação

fornecida pelo kit de robótica e esta por sua vez fazia a transmissão dos sinais através do infravermelho.

O protocolo de comunicação é o LNPd (*LegOs Networking Protocol daemon*). A interface deste protocolo é semelhante a do LNP (*Legos Networking Protocol*) que é um protocolo de troca de mensagens presente no *kernel* do sistema operacional brickOS [brickOS 2010]. Ele permite que dois ou mais RCX´s troquem mensagens de texto empacotadas. Os pacotes são codificados e transmitidos utilizando a tecnologia do infravermelho. Este esquema de comunicação esá representado pela Figura 4.3. A comunicação inicia-se assim que o botão *Iniciar Comunicação* é acionado, sendo feita então a verificação do protocolo LNPd.

Figura 4.3: Comunicação entre o computador e o robô (RCX)

As ferramentas que foram utilizadas para a implementação da interface desta versão foram a biblioteca gráfica QT e o *QT Designer*. O QT é uma classe da biblioteca C++ e o *Qt Designer* é uma caixa de ferramentas GUI bastante útil quando se deseja utilizar funções gráficas no ambiente Unix. Uma das principais características da biblioteca QT é o seu mecanismo de sinais e slots.

Sinais e slots são utilizados para comunicação entre objetos. O Qt *Designer* é uma ferramenta de desenvolvimento e implementação de interfaces para usuário que faz uso da biblioteca gráfica QT. Esse ambiente de desenvolvimento auxilia na organização de objetos em formulários, propiciando que se desenvolvam sistemas com *layouts* apropriados.

Nesta primeira versão do **RoboEduc** não foi feito nenhum estudo de casos de uso, assim como nenhum diagrama UML para implementação. Em contra partida, no desenvolvimento desta primeira etapa foi realizado o teste de usabilidade (Exploração) através de experimentos no qual podemos constatar que a experiência da implementação foi inovadora e satisfatória, o que nos estimulou a implementação de novas funcionalidades e no aperfeiçoamento do *layout*.

4.1.1 Versão 1.1

Neste novo protótipo do *software*, os pré-requisitos continuam os mesmos. Com o resultado do teste de usabilidade da etapa anterior foi constatado a necessidade de acrescentar uma nova funcionalidade, a de programação. Agora o *software* assiste na construção, controle e programação de diversos modelos de protótipos de robôs utilizando a mesma tecnologia LEGO.

Uma outra melhoria foi a implementação de um banco de dados de protótipos agrupados pelo nível de complexidade de construção e das tarefas que eles podem realizar, que na versão anterior não havia esta distinção e só existiam apenas 4 tipos de protótipos de robôs.

Para este protótipo do *software* foi feito um estudo de casos de uso na Linguagem de Modelagem Unificada (UML), que é responsável por definir e descrever os requisitos funcionais do *software*. Utilizaremos o diagrama da Figura 4.4 para explicar essas novas funcionalidades. Observe que o diagrama é separado em duas partes, cada uma delas representando as funcionalidades do *software* que se inter-relacionam com as entidades usuário do programa e protótipo do robô.

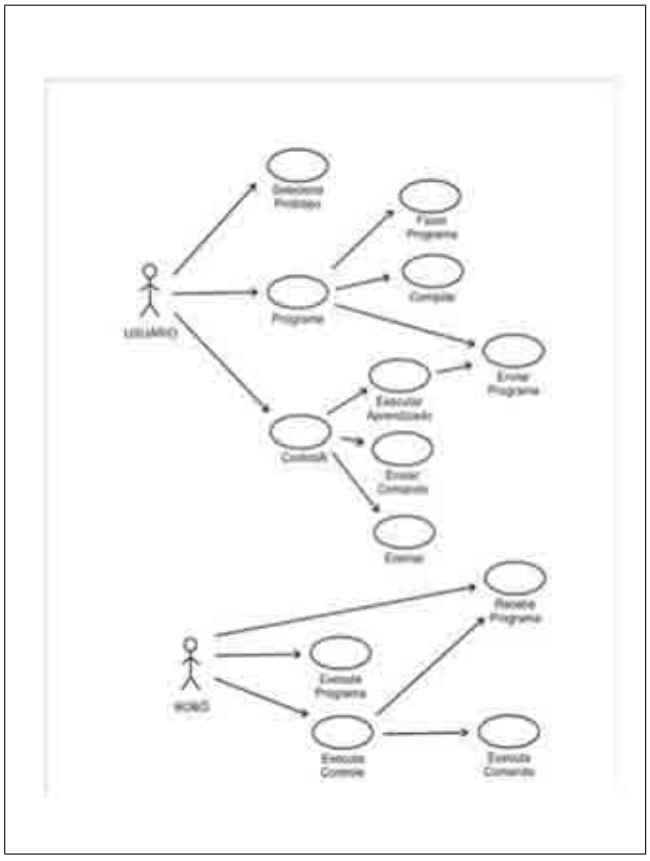

Figura 4.4: Diagrama de casos de uso

O usuário do *software* tem três casos de utilização do sistema:

1. Selecionar um protótipo;
2. Programar;
3. Controlar.

No Caso de Uso 1 (Selecionar um protótipo), o usuário dispõe de vários tipos de protótipos de robôs, que podem ser selecionados no lado direito inferior da tela principal. Os protótipos são agrupados pelo nível de complexidade em sua construção. Com isso, o usuário pode escolher entre protótipos básicos, intermediários e avançados. Ao selecionar um protótipo, o *software* exibe uma imagem do lado direto superior da tela e do lado esquerdo mostra sua descrição e as instruções de como construí-lo. Cada protótipo tem um conjunto de tarefas (jogos educacionais) associado para ser executado. Conforme a Figura 4.5.

Figura 4.5: Seleção do protótipo

No Caso de Uso 2 (Programar), o usuário programa um protótipo. Esta é uma funcionalidade disponível a partir desta versão e está representada na Figura 4.6.

Para isto é necessário escolher o nível de programação que se deseja utilizar. Estas opções encontram-se do lado direito inferior da tela principal. O nível de programação refere-se ao nível de abstração dos objetos disponíveis para a programação do robô.

Por exemplo, no nível básico o usuário dispõe simplesmente de botões que representam as ações de alto nível do robô, isto é, andar para frente ou girar à direita, porém as opções para editar, compilar ou enviar programa ao robô ficam transparentes. Basta simplesmente clicar no botão programar, que inicializa o processo de comunicação do *software* com o robô, para que seja enviado o programa (o *software* traduz os comandos em um programa para o compilador, depois é compilado e finalmente o programa é enviado ao robô utilizando a torre de comunicação infravermelho conectada na porta serial do computador).

Esta programação é feita pelo mecanismo de arrastar e soltar. Nos níveis mais avançados, é possível realizar comandos de programação baseados na linguagem de progra-

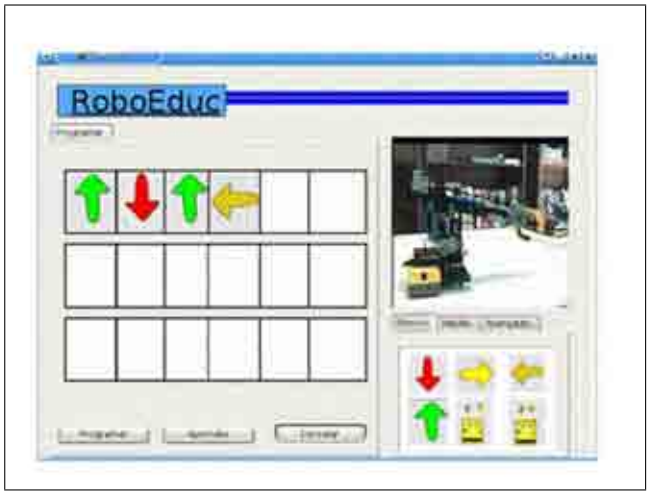

Figura 4.6: Funcionalidade programar

mação C. Porém as opções de compilação, depuração e transmissão dos programas não são transparentes ao usuário.

No Caso de Uso 3 (Controlar), já disponível na versão 1.0. O usuário controla remotamente um protótipo robótico enviando diretamente comandos ao robô para que sejam executados. Outra ação que pode ser realizada é que os comandos que são enviados ao robô podem ser gravados pelo *software* (ensinar ao robô). Quando gravados, os comandos são traduzidos automaticamente em programas que posteriormente poderão ser executados, permitindo assim que o usuário possa ver o comportamento do robô de forma autônoma. Esta funcionalidade do *software* **RoboEduc** pode ser entendida como ensino ao robô por parte do usuário, já que elee não está programando diretamente o robô. Os comandos disponíveis para controle do robô são os mesmos disponíveis na programação no nível básico.

Outro diagrama que foi feito para esta versão é o de classes, que consiste em mostrar as principais classes e suas relações. O *software* pode ser descrito com cinco classes e as relações entre elas são mostradas na Figura 4.7.

As cinco classes que compõem o *software* são: duas entidades representativas do mundo real (“Usuário” e “Robô”) e três elementos que representam o programa (“Comando”, “Programa” e “Protótipo”). A classe “Usuário” (user) representa o usuário no mundo real. O usuário pode interagir com o *software* de maneiras diferentes, como já foi explicado no diagrama de caso de uso.

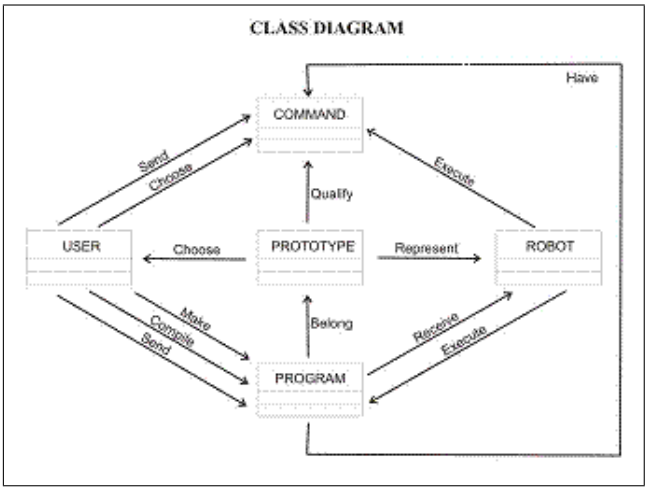

Figura 4.7: Diagrama de classes

Nível	Funções	Interface
0	Controle Remoto utilizando funcionalidades do robô	Visual
1	Programação utilizando elementos do controle remoto	Visual
2	Programação utilizando elementos do controle remoto e controle de fluxo	Visual
3	Programação utilizando elementos do controle remoto e controle de fluxo	Textual
4	Programação básica com elementos dos robôs (motores e sensores)	Visual
5	Programação Avançada com elementos do Robô	Textual

Tabela 4.1: Níveis de complexidade

4.2 Versão 2.0

Para versão 2.0 do *software* **RoboEduc**, não foram alterados os seus pré-requisitos de funcionamento. Porém, na ferramenta utilizada QT foi feito um *update* para o QT 4.0 e acrescentou-se uma nova linguagem o XML (eXtensible Markup Language) para a construção das interfaces com o usuário. As bibliotecas do BrickOS (SO e compilador) para as interfaces com os protótipos robóticos foram mantidas.

O **RoboEduc**, como já dito anteriormente, é um ambiente de programação voltado para educação e, portanto, especialmente adequado para o trabalho com crianças. Desde a sua criação, o **RoboEduc** contém características eminentemente gráficas. À medida que se utilizava a ferramenta, notou-se a necessidade de acrescentar mais recursos gráficos para torná-la mais acessível e mais apropriada para crianças do ensino infantil. Neste trabalho, procura-se dar atenção a ambos os aspectos do **RoboEduc**: os aspectos gráficos e os aspectos relacionados a manipulação de palavras reservadas da linguagem e o seu processamento. Nesta versão foram formalizados os níveis de complexidade para programação. Estes níveis estão representados pela Tabela 4.1

Todos esses níveis já estavam disponíveis na versão anterior deste protótipo. Para

esta nova versão do protótipo do *software* **RoboEduc**, considerou-se a possibilidade de tradução dos programas de um nível inferior a um nível superior de complexidade. Isto certamente ajudará o usuário a se familiarizar com os conceitos do novo nível relacionando os mesmos com o anterior.

O XML foi utilizado para permitir que o *software* tenha capacidade de expansão automática do Caso de Uso 1 descrito na versão 1.1 (Selecionar Protótipo). Isto é, o *software* não terá que ser compilado novamente ou modificado na hora de acrescentar protótipos ao mesmo, além disso, um usuário mais experiente poderá fazer esta tarefa com muita facilidade. Agora, as funcionalidades, os modelos e protótipos são especificados utilizando XML.

Estas especificações incluem as principais características dos mesmos e o endereço onde se encontram as imagens que representam as diferentes partes dos protótipos, modelos e funcionalidades. Isto permite que o usuário possa personalizar suas imagens dos diferentes elementos do *software*.

O *software* carrega automaticamente os elementos contidos nos arquivos XML. Por exemplo, no caso das funcionalidades, o código XML deve incluir o nome, que é um identificador único para o *software*, além disso, deve incluir a imagem que o representa e como os motores serão acionados, isto é, se para frente, para trás, ou ainda se ficam parados.

As interfaces com o usuário sofreram alterações. A Figura 4.8 mostra os elementos que representam os modelos do robôs.

A Figura 4.9 exibi os componentes dos protótipos, ou seja, os protótipos construídos. As suas funcionalidades são carregadas em tempo de execução, levando em consideração a informação contida nos arquivos XML do *software*.

A tela de controle remoto está ilustrada na Figura 4.10 e o primeiro nível de programação na Figura 4.11.

4.2.1 Versão 2.1

Neste novo protótipo do *software*, os pré-requisitos continuam os mesmos. Com o resultado do teste de usabilidade da etapa anterior foi constatado a necessidade de modificações na interface com o usuário. Foram alterados as cores e alguns ícones.

Para esta versão basicamente, além de controlar um robô, o aluno tem a opção de armazenar os comandos executados em um arquivo para executá-lo novamente. Pode também abrir um arquivo já gravado anteriormente e executá-lo novamente.

As Figuras 4.12, 4.13, 4.14 mostram as telas que sofreram alterações.

4.3 Versão 3.0

Na terceira versão o *software* foi totalmente reformulado, principalmente na sua interface [Barros 2008]. A equipe de desenvolvimento, através dos seus experimentos, observou a necessidade de optar por algo mais infantil, ficando, mas evidente que o *software* foi desenvolvido para contemplar alunos desde séries iniciais. Outro grande avanço desta

Figura 4.8: Escolha do modelo

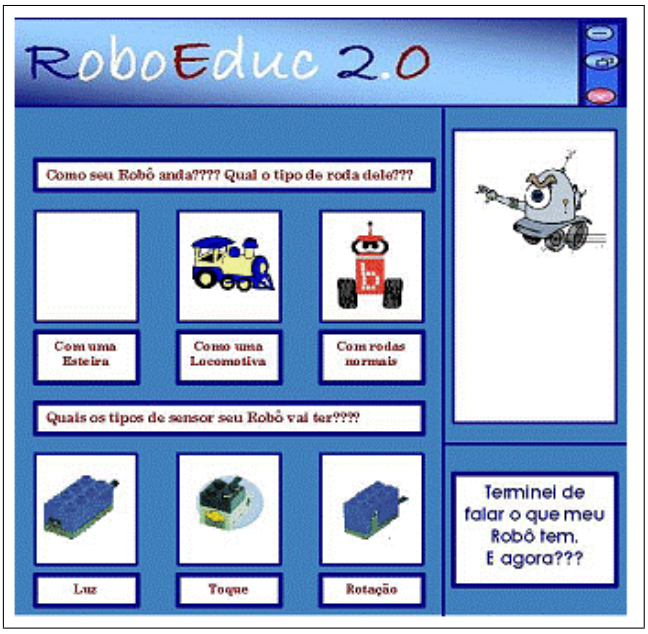

Figura 4.9: Escolha de componentes

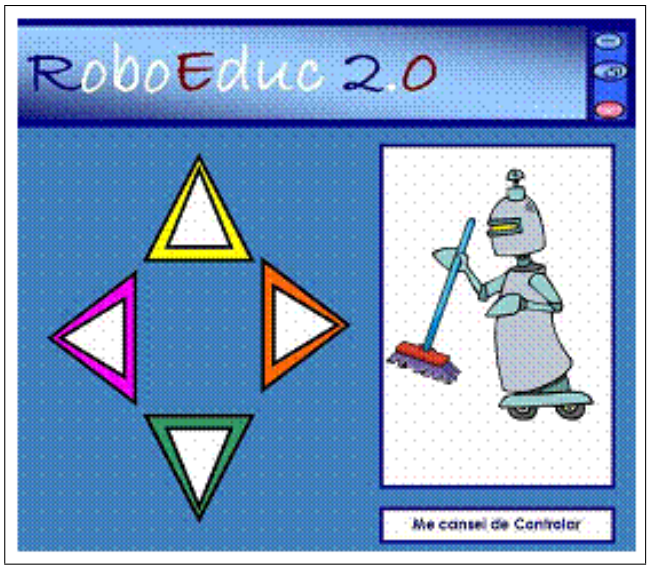

Figura 4.10: Controle remoto

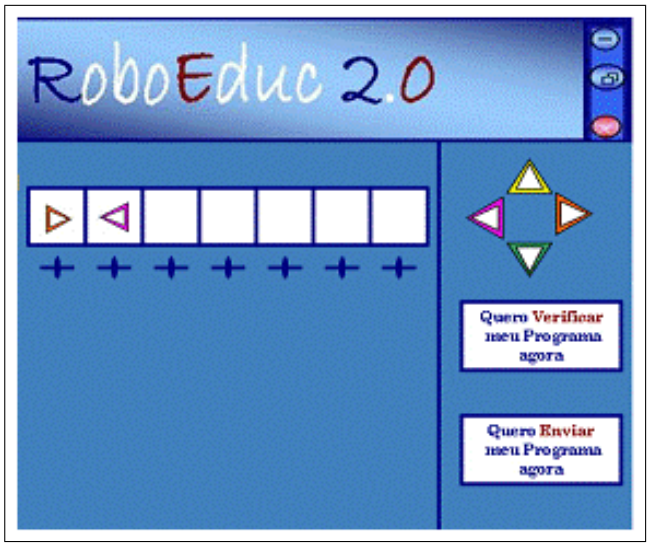

Figura 4.11: Nível de programação 1

Figura 4.12: Tela inicial

Figura 4.13: Controlar

Figura 4.14: Ensinar

versão é o módulo Autoria. Este módulo é destinado a professores/monitores (alunos ou não) de robótica educacional.

Esta nova funcionalidade foi resultado dos testes de exploração com os usuários do *software*. O resultado dos testes apontou que os professores necessitavam de um ambiente integrado que auxiliasse no planejamento e organização das aulas.

Neste módulo, o professor pode projetar o robô através da criação de modelos, bases, atuadores, sensores, ações e definir um conjunto de tarefas (jogos educacionais) a serem executadas pelo robô e pelos alunos.

É possível também construir um protótipo que pode ser manipulado através do controlar, ensinar e programar em cinco níveis de complexidades diferentes.

Nesta terceira versão, na tela inicial do **RoboEduc**, o aluno já visualiza esta distinção entre os dois módulos Aluno e Autoria. Figura 4.15.

A interação com o usuário ocorre por meio de uma interface simples, estimulante e atrativa, com muitas cores e desenhos, que, na realidade, consiste de artifícios para proporcionar ao usuário o aprendizado de conceitos complexos, como, por exemplo, técnicas de programação. O usuário pode acessar o *software* de duas maneiras: como professor, que pode acessar todas as funcionalidades, e como aluno, onde o seu acesso é restrito.

As funcionalidades podem ser executadas através do teclado ou do *mouse*, proporcionando uma familiarização com esses dois tipos de periféricos. A comunicação entre o *software* e o robô é realizada por uma torre infravermelha conectada ao computador pela porta USB, e o protocolo de comunicação utilizado é o LNP.

Para o desenvolvimento do *software* foi utilizada a linguagem de programação C++ e as interfaces com o usuário foram programadas com a biblioteca Qt 4.3. Os arquivos de conteúdo estão no formato XML e os arquivos para a manipulação dos robôs foram

Figura 4.15: Tela inicial do **RoboEduc 3.0**

programados na linguagem C, tendo como biblioteca de interface com o hardware o BrickOS. As ferramentas utilizadas para a implementação do *software* foram o compilador GCC, a biblioteca gráfica Qt 4.3, o compilador BrickOS e o Eclipse, que é uma IDE *(Integrated Development Environment)* que reúne características e ferramentas de apoio ao desenvolvimento de um *software*.

A metodologia utilizada no desenvolvimento desta versão continua a mesma das versões anteriores, a prototipagem evolucionária.

4.3.1 Visões UML

As visões UML são utilizadas para descrever os diferentes aspectos do *software*, representam uma abstração formada por um conjunto de diagramas. A partir de um conjunto de visões é possível realizar uma descrição completa do sistema a ser constituído.

As visões elaboradas para o **RoboEduc 3.0** foram:

1. Visão de Casos de Uso: mostra a funcionalidade do sistema do ponto de vista externo. É representada por um conjunto de atores que interagem com o sistema, sendo descrita pelos diagramas de casos de uso.
2. Visão Lógica: Descrevem a organização do sistema, os módulos principais, os relacionamentos entre os módulos e suas funcionalidades. A visão Lógica envolve a estrutura estática do sistema (módulos, casses, objetos, relacionamentos) representada pelo Diagrama de Classes.

4.3.2 Diagramas UML: Caso de Uso

As funcionalidades do *software* podem ser visualizadas nos diagramas mostrados nas Figuras 4.16 e 4.17. Nestas figuras, é possível observar as ações que o usuário pode executar no *software*. Cada *Caso de Uso* será descrito a seguir.

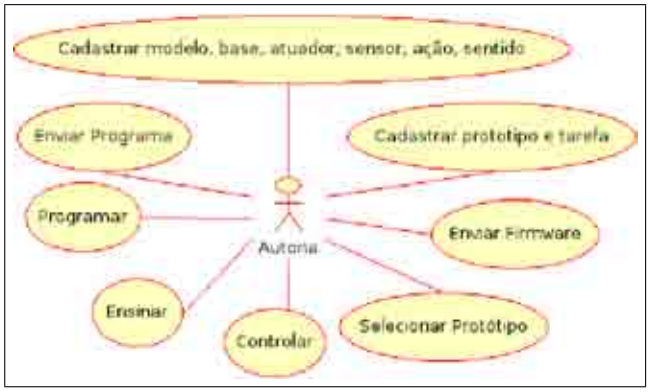

Figura 4.16: Diagrama de caso de uso do *ator autoria*

Figura 4.17: Diagrama de caso de uso do *ator aluno*

Selecionar módulo aluno ou módulo autoria

Na tela principal do **RoboEduc 3.0**, Figura 4.15, é disponibilizado ao usuário o módulo *Aluno* e o módulo *Autoria*. No módulo *Autoria* o usuário pode:

1. *Cadastrar modelo, base, atuador, sensor, ação, sentido.*;
2. *Cadastrar protótipos e tarefas*;
3. *Enviar Firmware para o robô*;
4. *Selecionar protótipos cadastrados;*

5. *Controlar protótipo;*
6. *Ensinar protótipo;*
7. *Programar protótipo;*
8. *Compilar programas;*
9. *Enviar programas.*

No módulo *Aluno* o usuário pode:

1. *Enviar Firmware para o robô;*
2. *Selecionar protótipos cadastrados;*
3. *Controlar protótipo;*
4. *Ensinar protótipo;*
5. *Programar protótipo;*
6. *Compilar programas;*
7. *Enviar programas.*

Enviar *firmware*

Neste caso de uso, o protocolo de comunicação, entre o robô e o computador, é inicializado quando o usuário clica no botão *Firmware*. O *firmware* é um sistema operacional que possui uma lista de prioridades para alocação de tarefas (*preemptive*), que controla diretamente o hardware. A comunicação é realizada através do protocolo LNP (protocolo de troca de mensagens presente no núcleo (*kernel*) do sistema operacional do BRICKOS). A transmissão física ocorre usando uma torre de comunicação infravermelha conectada ao computador pela porta USB.

Cadastrar modelo, base, atuador, sensor, ação e sentido

O caso de uso *Cadastrar modelo, base, atuador, sensor, ação e sentido* apresenta a tela na qual são exibidos os modelos já cadastrados. O usuário *Autoria*, ao clicar no botão *Criar*, pode cadastrar o nome e a descrição e associar uma imagem do tipo *gif animado* ao novo modelo. Isso é realizado também para bases, atuadores e sensores. Na criação das ações e dos sentidos, além dos itens anteriores, também é cadastrado o conteúdo das ações e dos sentidos. Existe nas telas de criação a opção de construir o manual de cada item novo. As suas interfaces podem ser vista nas Figuras 4.18 e 4.19.

Cadastrar protótipos e tarefas

Neste caso de uso, o usuário *Autoria* seleciona um modelo, uma base, um atuador, um sensor, uma ou mais ações, um ou mais sentidos e cadastra um nome para o protótipo e uma descrição em linhas gerias, além de associar uma imagem do tipo *gif animado* para o protótipo. Existe também o cadastro de tarefas, que são associadas ao protótipo que está sendo montado. Após escolher todas as opções, o usuário clica no botão *Protótipo* e o *software* salva um arquivo com as informações referentes ao protótipo criado. A sua interface é ilustrada na Figura 4.20.

Figura 4.18: Criar um novo modelo, base, atuador, sensor.

Figura 4.19: Criar uma nova ação

Selecionar protótipos cadastrados

Neste caso de uso, o *software* disponibiliza ao usuário *(Aluno e Autoria)* os modelos de robôs que foram previamente cadastrados no caso de uso *Cadastrar modelo, base,*

Figura 4.20: Criar um novo protótipo

atuador, sensor, ação e sentido. Após escolher um modelo, são apresentadas ao usuário as opções de construção para determinado protótipo cadastrado pelo professor e, então, o usuário escolhe os componentes existentes no protótipo montado. Após esta etapa, será apresentada ao usuário uma tela com a descrição, a tarefa e uma imagem do tipo *gif animado* do protótipo escolhido. A próxima etapa é enviar o programa de escuta ao robô, sendo estabelecida a comunicação entre robô e o computador. A sua interface é ilustrada na Figura 4.21.

Controlar protótipo

No caso de uso *Controlar Protótipo*, as ações que esse protótipo pode executar são exibidas. Será inicializado o protocolo de comunicação entre o robô e o computador. Cada opção representa uma ação que foi previamente cadastrada para o protótipo selecionado. Esses comandos são enviados através do protocolo de comunicação ao robô. A sua interface é ilustrada na Figura 4.22.

Ensinar protótipo

Este caso de uso aborda, além das funcionalidades descritas no caso de uso *Controlar protótipo*, uma opção para armazenar os comandos executados em um arquivo e torná-lo apto a ser utilizado posteriormente. Outra funcionalidade deste caso de uso é abrir um arquivo existente e executá-lo. A tradução dos comandos e o envio do programa ao robô são feitos internamente pelo *software*, de forma transparente ao usuário. A sua interface é ilustrada na Figura 4.23.

Figura 4.21: Selecionar protótipo cadastrado

Figura 4.22: Controlar protótipo

Programar protótipo

Neste caso de uso, uma tela com os cinco níveis de programação possíveis do **Robo-Educ** é exibida ao usuário. A sua interface esta ilustrada na Figura 4.24.

O nível de programação refere-se ao nível de abstração dos objetos disponíveis para a programação do robô. Os níveis de programação são:

Figura 4.23: Ensinar protótipo

Figura 4.24: Escolha dos níveis

- **Nível 1**: As funções são semelhantes ao caso de uso *Controlar protótipo*. A programação é realizada no modo gráfico por meio do mecanismo arrastar e soltar. As funcionalidades deste nível são: *Compilar programa, Enviar programa, Salvar programa, Abrir programa*. A tradução da linguagem gráfica para a linguagem de BrickOS é transparente ao usuário. A interface deste nível é representada na Figura 4.25

Figura 4.25: Nível 1

- **Nível 2**: Neste nível as funções são semelhantes ao nível anterior. São acrescentadas as estruturas de fluxos. A programação neste nível também é realizada de modo gráfico por meio do mecanismo arrastar e soltar. As funcionalidades são: *Compilar programa, Enviar programa, Salvar programa e Abrir programa*. A tradução da linguagem gráfica para a linguagem de *BrickOS* é transparente ao usuário. A interface deste nível este representada na Figura 4.26

Figura 4.26: Nível 2

- **Nível 3**: Este nível tem as funções semelhantes ao nível 2, porém a programação é realizada no modo texto. A linguagem de programação utilizada é **"RoboEduc"**,

que foi desenvolvida em conjunto com o *software* **RoboEduc**. A interface com o usuário é um editor de texto semelhante aos editores de outras linguagens existentes. A tradução da linguagem **RoboEduc** para a linguagem de *BrickOS* é transparente ao usuário. As funcionalidades são: *Utilidades do editor de texto, Compilar programa, Enviar programa, Salvar programa, Abrir programa*. A interface deste nível está representada na Figura 4.27

Figura 4.27: Nível 3

- **Nível 4**: Neste nível, as funções são semelhantes ás do RoboLab e sua programação é realizada em modo gráfico por meio do mecanismo arrastar e soltar. A tradução da linguagem gráfica para o BrickOS é transparente para o usuário. As funcionalidades são: *Compilar programa, Enviar programa, Salvar programa, Abrir programa*. A interface deste nível está representada na Figura 4.28
- **Nível 5**: Neste nível as funcionalidades são as funções do BrickOS. A programação é realizada no modo texto. A interface com o usuário é um editor de texto semelhante aos editores de outras linguagens existentes. As funcionalidades são: *Compilar programa, Enviar programa, Salvar programa e Abrir programa*. A interface deste nível está representada na Figura 4.29

Compilar programa

O caso de uso *Compilar programa* ocorre quando o usuário já escreveu o seu programa e deseja saber se existe algum erro de compilação. Como mencionado no item anterior, existem vários níveis de programação, portanto, a compilação depende de qual nível o programa foi escrito. Para compilar, o usuário deve apertar o botão *Compilar*, presente na tela. O processamento ocorre segundo as seguintes etapas:

- Análise do código-fonte na linguagem **RoboEduc** através do tradutor desenvolvido junto com o *software* **RoboEduc**;

Figura 4.28: Nível 4

Figura 4.29: Nível 5

- Verificação de erros;
- Exibição de tela mostrando se existe erro ou não;
- Geração do código-fonte em linguagem C, caso o nível escolhido seja diferente do nível 5;
- Verificação da existência de erros;
- Exibição de tela mostrando se existe erro ou não.

Enviar comando

O caso de uso *Enviar comando* representa o envio de pacotes através do protocolo de comunicação usado no processador do robô (o RCX). Para que tais pacotes sejam enviados, torna-se necessário o clique no botão referente à ação correspondente.

Enviar programas

Este caso de uso diz respeito ao ato de enviar o programa já compilado e verificado para o RCX. Implicitamente, este caso de uso faz também a inicialização e verificação do protocolo de comunicação.

4.4 Implementação do RoboEduc 3.0

Este *software* é bastante dinâmico. A partir da modelagem adotada, quando é necessário adicionar qualquer conteúdo ao *software*, esse conteúdo é armazenado em um arquivo XML. Portanto, não existe a necessidade de recompilar o programa. A estrutura do arquivo XML funciona como um repositório de dados, sendo mostrada na Figura 4.30.

O arquivo XML é lido na tela inicial de cada módulo *Aluno e Autoria*, uma vez que este contém as informações dos nomes, das descrições dos componentes, dos endereço das imagens animadas dos modelos, das bases, dos atuadores e de todos os outros componentes, como também de todas as funcionalidades cadastradas. Ao ler este arquivo XML, a classe escrita em Qt faz o uso do mecanismo *DOM*. Este mecanismo proporciona que as informações sejam extraídas do XML e armazenadas em uma árvore na memória. A partir desta árvore, as informações são armazenadas em um objeto da classe *QList*. Para organizar as informações em listas, foram criadas classes com os campos necessários e os tipos das listas foram definidos de acordo com o tipo de objeto necessário.

Todas as classes que contêm imagens tipo *gif* ou informações referentes ao conteúdo do *software* foram criadas através da manipulação de arquivos XML. Essa característica torna o programa um *software* de Autoria, facilmente extensível e dinâmico. O *software* contém uma linguagem de programação própria, denominada **RoboEduc**, cujos programas em forma textual possuem terminação *.rob*. A necessidade da criação desta linguagem ocorreu pela complexidade encontrada nas linguagens de programação já existentes.

Como mencionado no Capítulo 2, em virtude das linguagens de programação possuírem dois segmentos (Compilação e Interpretação), a linguagem **RoboEduc** foi desenvolvida para englobar esses dois módulos. Nos seus primeiros módulos ("Controlar"e o "Ensinar"), a linguagem **RoboEduc** é uma linguagem interpretada. Nos módulos de programação, a linguagem **RoboEduc** é compilada, uma vez que para cada estágio de aprendizado, a linguagem se adapta ao usuário. Este tipo de adaptação não é encontrado em outras linguagens. Entende-se que isso é o ideal em ambiente de aprendizagem uma vez que as nossas experiências nos mostraram que inicialmente o usuário não deve estar preocupado com a forma de escrever um programa e sim focar-se na sua lógica. Após adquirir certa maturidade, ele deve se ter atenção com a forma de escrita do programa.

```
<?xml version="1.0" encoding="UTF-8"?>
<RoboEduc>
  <modelo>
      <nome>Corredor</nome>
      <imagem>figura/modelos/corredorPrototipo.gif</imagem>
      <descricao>essa e a descricao do corredor</descricao>
      <base>
            <gif>figura/modelos/corredorBase.gif</gif>
            <nome>Base do Corredor</nome>
            <descricao>essa eh a descricao da base</descricao>
      </base>
      <atuador>
            <gif>figura/modelos/naopossui.gif</gif>
            <nome>NaoPossui</nome>
            <descricao>Este modelo NaoPossui atuador</descricao>
      </atuador>
      <sensor>
            <gif>figura/modelos/naopossui.gif</gif>
            <nome>NaoPossui</nome>
            <descricao>Este modelo NaoPossui sensor</descricao>
      </sensor>
      <acao>
      <gif>figura/setaCima.gif</gif>
      <nome>Frente</nome>
      <descricao>O robo se movimenta para a frente</descricao>
      <composicao>
            <item>motor_a_dir(fwd)</item>
            <item>motor_c_dir(fwd)</item>
      </composicao>
      </acao>
      <sentidos>
            <gif>figura/setaCima.gif</gif>
            <nome>NaoPossui</nome>
            <descricao>Este robo NaoPossui sentidos</descricao>
            <composicao>
                  <item>'sensorA luz'</item>
            </composicao>
      </sentidos>
</modelo>
```

Figura 4.30: Estrutura do arquivo XML

Para a implementação desta linguagem, foi desenvolvida uma gramática própria da linguagem, descrita a seguir.

4.4.1 Gramática da linguagem

As produções são as seguintes:

- *<tarefa> := tarefa <nome da tarefa> inicio <instrucao>* <fim>*
- *<nome da tarefa>:= letra(letra|numero)**

- *<controle de fluxo>:= <condicional> || <repeticao>*
- *<condicional>:= se <condicao> então <instrucao>* [senão <instrucao>*] fimse*
- *<condicao>:= (<variavel> <op> <numero>) || ((<variavel> <op> <numero>) <op logico> (<variavel> <op> <numero>)) || (não (<variavel> <op> <numero>)) || ((não (<variavel> <op> <numero>)) <op logico> (<variavel> <op> <numero>)) || ((<variavel> <op> <numero>) <op logico> (não (<variavel> <op> <numero>))) || ((não (<variavel> <op> <numero>)) <op logico> (não(<variavel> <op> <numero>)))*
- *<variavel>:=letra(letra|numero)**
- *<acao>:= executar <nome da tarefa> || <nome de acao> <numero> segundos*
- *<repeticao>:= <repita para> || <enquanto>*
- *<repita para>:= para <variavel> := <numero> ate <numero> faca <instrucao>* fimpara*
- *<enquanto>:=enquanto <condicao> faca <instrucao>* fimEnquanto*
- *<op>:= = || > || >= || < || <=*
- *<op logico>:= e || ou*
- *<letra>:=[A-Za-z]*
- *<instrucao>:= <controle de fluxo> || <acao>*
- *<nome de acao>:= letra(letra|numero)**
- *<numero>:=0||1||2||3||4||5||6||7||8||9**

4.4.2 As palavras reservadas da linguagem

As palavras reservadas usadas pela nossa linguagem são mostradas na Figura 4.31. Como podemos observar, são poucas palavras usadas pela linguagem, mostrando assim um alto nível de abstração. Apesar de ter um nível bem próximo da linguagem infantil, esta linguagem permite ensinar estruturas algorítmicas, tais como início de programa, fim de programa, controle de fluxo, comandos condicionais, chamadas de funções (elementares) e outras estruturas mais simples, como o envio de comandos simples.

4.4.3 O interpretador

A partir da definição da gramática acima, a etapa seguinte foi a implementação do interpretador da linguagem **RoboEduc** para a linguagem C e BrickOS. As etapas da implementação foram as seguintes:

- Implementação do Analisador Léxico;
- Implementação do Analisador Sintático.

No analisador léxico, a entrada é um arquivo *.rob* na linguagem **RoboEduc** que separa o arquivo em *tokens*. No analisador sintático, os *tokens* são recebidos do analisador léxico e é analisada a estrutura do arquivo a fim de verificar se está de acordo com as regras da gramática definida. Uma vez analisado sintaticamente, é iniciado o processo de interpretação da linguagem **RoboEduc** para linguagem C e BrickOS. A saída do analisador sintático é um arquivo escrito em C e BrickOS. Após esta etapa, faz-se uma chamada

Palavras Reservadas	
tarefa	repita
inicio	enquanto
fim	fimenquanto
se	para
entao	ate
fimse	faca
senao	fimpara
fimsenao	e
executar	ou
nao	

Figura 4.31: As palavras reservadas da linguagem **RoboEduc**

de sistema para o compilador BrickOS, que compila o arquivo C. A saída do compilador BrickOS é um arquivo *.lx* que é enviado ao robô. Todas as etapas de tradução reportam mensagens de erros caso eles ocorram.

4.4.4 Diagramas UML: Classe

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam os objetos que são gerenciados pela aplicação modelada. Classes podem se relacionar com outras de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares). Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações. O diagrama de classes é considerado estático já que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema. Um sistema normalmente possui alguns diagramas de classes, já que não são todas as classes que estão inseridas em um único diagrama e certa classe pode participar de vários diagramas de classes. O diagrama de classes mostra uma representação conceitual das informações, que abrangem uma coleção de elementos declarativos válidos nos diversos estágios do sistema. A implementação do *software* **RoboEduc** seguiu o seguinte diagrama de classes.

A Figura 4.32 ilustra o diagrama de classes construído para o **RoboEduc**. Este diagrama representa uma visão geral dos dados através das classes principais. Cada janela do *software* foi implementada como uma classe. Foge do escopo deste texto a descrição detalhada de cada classe. Assim, colocamos basicamente as classes de manipulação do XML e do interpretador, que são as mais importantes. Outras classes, como de interface

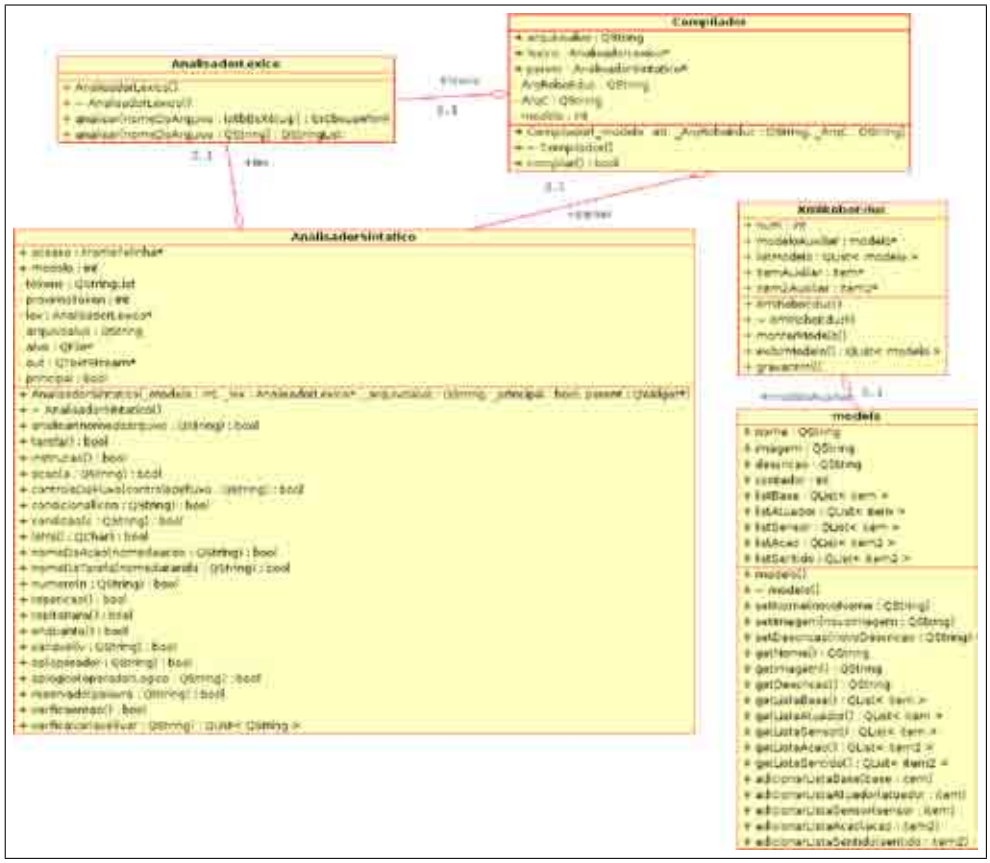

Figura 4.32: Diagrama das principais classes do **RoboEduc**

e interação com o usuário, fazem parte do Qt 4.3. Olhando a Figura 4.32, podemos perceber que as classes sintetizam as funcionalidades descritas acima, nos casos de uso. Nos experimentos, serão mostradas funcionalidades práticas, com a execução dos métodos definidos nessas classes.

4.5 Versão 4.0

Esta nova versão implementa os resultados obtidos após a análise dos resultados dos testes de usabilidade que foram descritos no relatório final Anexo G.

Para o desenvolvimento desta nova versão foram mantidas as mesmas ferramentas utilizadas nas versões anteriores. Os requisitos de funcionamento não sofreram alterações. O sistema de transmissão de dados entre o *software* e o robô passou a ser via *Bluetooth*.

Uma das novidades desta versão é o simulador da linguagem **Educ** e o ambiente de projetos que foram incorporados ao *software*.

O relatório propõe uma reformulação no *layout* das telas. Foram definidas as áreas de *menus*, *submenus*, as áreas de trabalho, localizações dos títulos e a área de ajuda *off-line*. Para uma melhor compreensão das funcionalidades do *menu* os ícones foram redesenhados.

A Figura 4.33 ilustra essas modificações. Nesta tela inicial foram implementadas três funcionalidades para o usuário Aluno: Projetar, Montar e Programar.

Figura 4.33: Tela inicial do **RoboEduc 4.0**

No projetar o aluno é direcionado para tela de projetos que está ilustrada na Figura 4.34. Este módulo funciona como um carimbo. O aluno clica em algum componente que está na janela Ferramentas e depois é só clicar na área de trabalho. Na janela Ferramentas colocamos as principais peças estruturais que compõem um robô, assim como algumas opções de desenho a mão livre como um círculo, uma reta etc.

No Montar, o aluno tem a possibilidade de escolher quais componentes farão parte do seu robô. A Figura 4.35 ilustra esta tela. Esta tela não sofreu nenhuma alteração funcional em relação a versão anterior. As modificações foram feitas no seu *layout* e foi trocado o nome da aba Modelo por Categoria para ficar mais intuitivo.

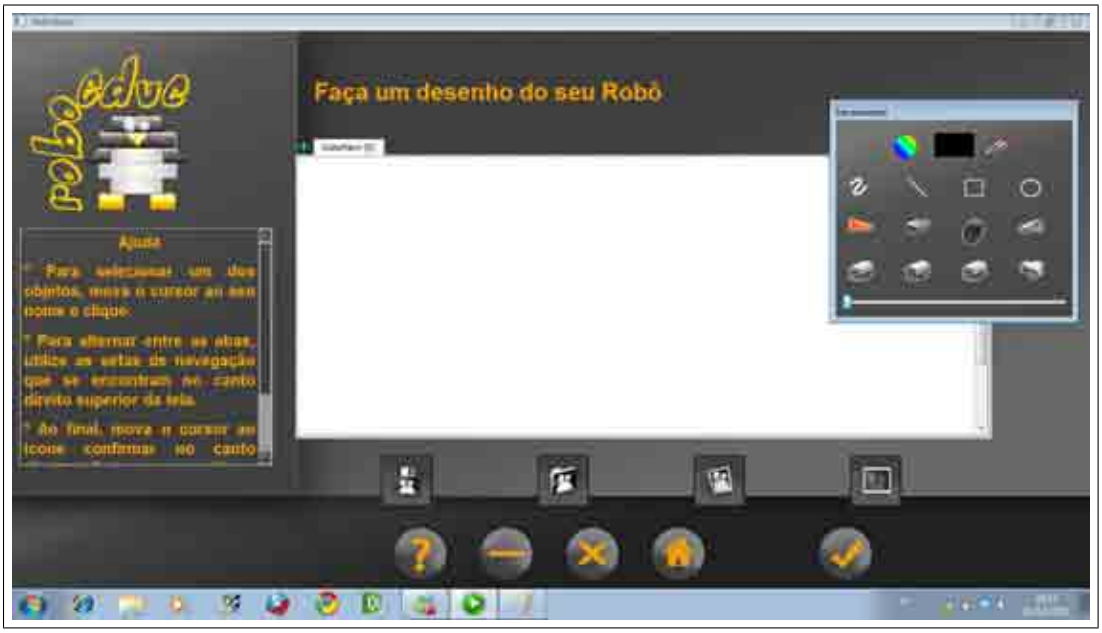

Figura 4.34: Projetar

Figura 4.35: Montar robô

Por fim, na opção Programar, o Aluno é direcionado ao banco de dados do *software* que está ilustrado da Figura 4.36, onde ele tem a possibilidade de escolher um robô já cadastrado e depois realizar a tarefa que foi programada pelo professor. Esses robôs foram previamente cadastrados pelo professor para serem utilizados em aulas posteriores. Nesta tela foi implementada também a funcionalidade de deletar protótipo.

Figura 4.36: Banco de dados

A tela que ilustra a descrição do robô montado ou escolhido através do banco de dados é ilustrada na Figura 4.37.

Figura 4.37: Descrição do robô e a tarefa a ser realizada

Os módulos Controlar e Ensinar apenas sofreram alterações nas suas interfaces, como podem ser vistas nas Figuras 4.38 e 4.39, respectivamente.

Figura 4.38: Tela do controlar

Figura 4.39: Tela do ensinar

O relatório propôs também um aprimoramento da linguagem de programação e um novo nome para diferencia-lá do *software*. A linguagem foi intitulada de **Educ**, que é uma implementação brasileira, em português, da linguagem de baixo nível NXC [NXC 2011], que é voltada, especialmente, para uso da robótica na educação.

A partir desta linguagem de programação que foi concebida, desenvolvida e direci-

Figura 4.40: Escolha dos níveis

onada a aplicações educacionais, o *software* **RoboEduc** naturalmente incorporará uma metodologia inovadora na educação.

Neste novo protótipo acrescentamos novas funcionalidades ao módulo Autoria, um aprimoramento da linguagem textual para suportar a interação com sensores de diversos tipos e uma atualização na linguagem de baixo nível que se comunica com o dispositivo robótico.

Preocupamo-nos também em adequar as funcionalidades do *software* com o embasamento pedagógico sobre os níveis de programação do *software* **RoboEduc**, que está descrito no Capítulo 3 na Seção 3.2. A Figura 4.40 ilustra a escolha dos níveis de programação.

A gramática não sofreu nenhuma alteração nas suas regras.

As Figuras 4.41 e 4.42 ilustram os níveis 1 e 3, os demais níveis seguem o mesmo estilo de interface.

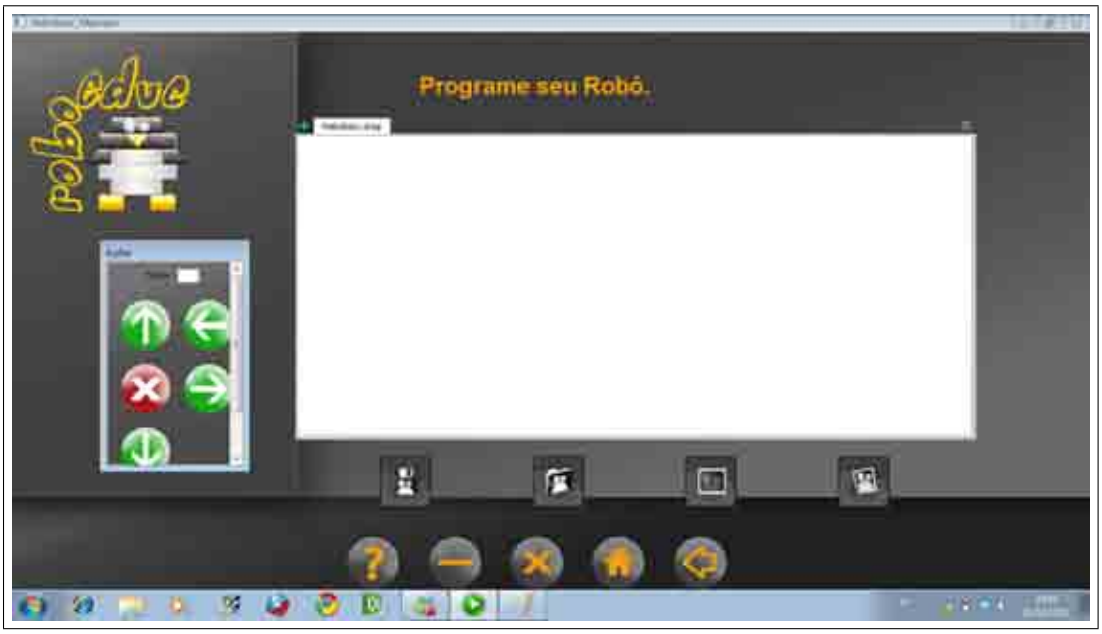

Figura 4.41: Nível 1 do **RoboEduc 4.0**

Figura 4.42: Nível 3 do **RoboEduc 4.0**

4.5.1 Experimento do *RoboEduc 4.0*

Para ilustrar os novos *layouts* e todas as funcionalidades que foram implementadas, será descrito uma experiência que foi realizada com o *software* onde neste grupo foram constatados alunos dos três níveis da sociedade. Esta divisão foi feita através de critérios estabelecidos pelo docente responsável pelo grupo.

Este experimento foi realizado em uma escola da periferia da cidade. De acordo com o plano de aula que foi estabelecido para esta oficina, foram utilizados todos os módulos e níveis da linguagem de programação **RoboEduc**. O tema gerador da oficina é "Encontrando o Caminho", e o tempo estipulado para realização das atividades foi de duas horas. Os recursos didáticos utilizados foram: 3 computadores com sistema operacional *Windows* com o *software* **RoboEduc 4.0**, 3 Kits LEGO, 3 mapas do Rio Grande do Norte (marcando as cidades do Natal, Mossoró, Caicó, Touros, Santa Cruz, Paus Ferros, Macau e Martins), 24 bandeirinhas com nomes das cidades, 3 canetas, e 3 folhas impressas para registro das conclusões da atividade. O mapa da atividade é mostrado na Figura 4.43.

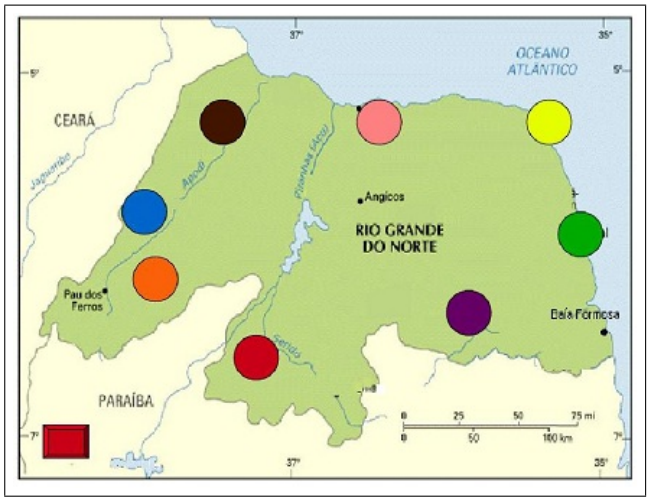

Figura 4.43: Mapa do RN

Os objetivos específicos desta oficina foram: o projeto do robô utilizando o *software* **RoboEduc** e a montagem em grupo de um robô móvel com o kit Lego. O robô foi composto por três motores, um para cada par de rodas e um para a caneta. No primeiro momento da oficina foi utilizado o módulo de projetos do *software*, que está ilustrado na Figura 4.44A. Neste módulo do *software* são disponibilizadas todas as ferramentas necessárias para o projeto dos robôs, e o seu funcionamento é simples e intuitivo. A interface deste módulo funciona de forma similar a um carimbo. O aluno clica na imagem desejada e depois clica na área de projeto.

Após esta etapa, cada grupo montou o robô projetado com um kit de robótica da LEGO e em seguida manipulou o robô construído através do *software* **RoboEduc 4.0**, observando as noções de lateralidade, distância, velocidade e posicionamento.

Nesta etapa da tarefa foram utilizados todos os módulos do *software*, cada grupo respeitando o seu grau de conhecimento. No grupo identificado como analfabetos digitais,

foi utilizado o módulo de controle para execução da atividade, cuja tela está ilustrada na Figura 4.44B.

Para o grupo identificado como migrantes digitais, foi utilizado o módulo de ensinar que está ilustrado na Figura 4.44C. Este módulo tem por objetivo armazenar as ações que os alunos selecionaram para realizar o desafio proposto. Neste módulo já é necessário certo grau de lógica para prever o que o robô deveria fazer (qual a trajetória a ser seguida).

O terceiro grupo, identificado como nativos digitais, utilizou os demais níveis que a linguagem oferece. Ressaltando que o último nível de programação foi utilizado com a ajuda dos monitores de robótica, por se tratar de uma linguagem mais complexa. A tela do programa com as atividades realizadas por este grupo está ilustrada na Figura 4.44D, E, F.

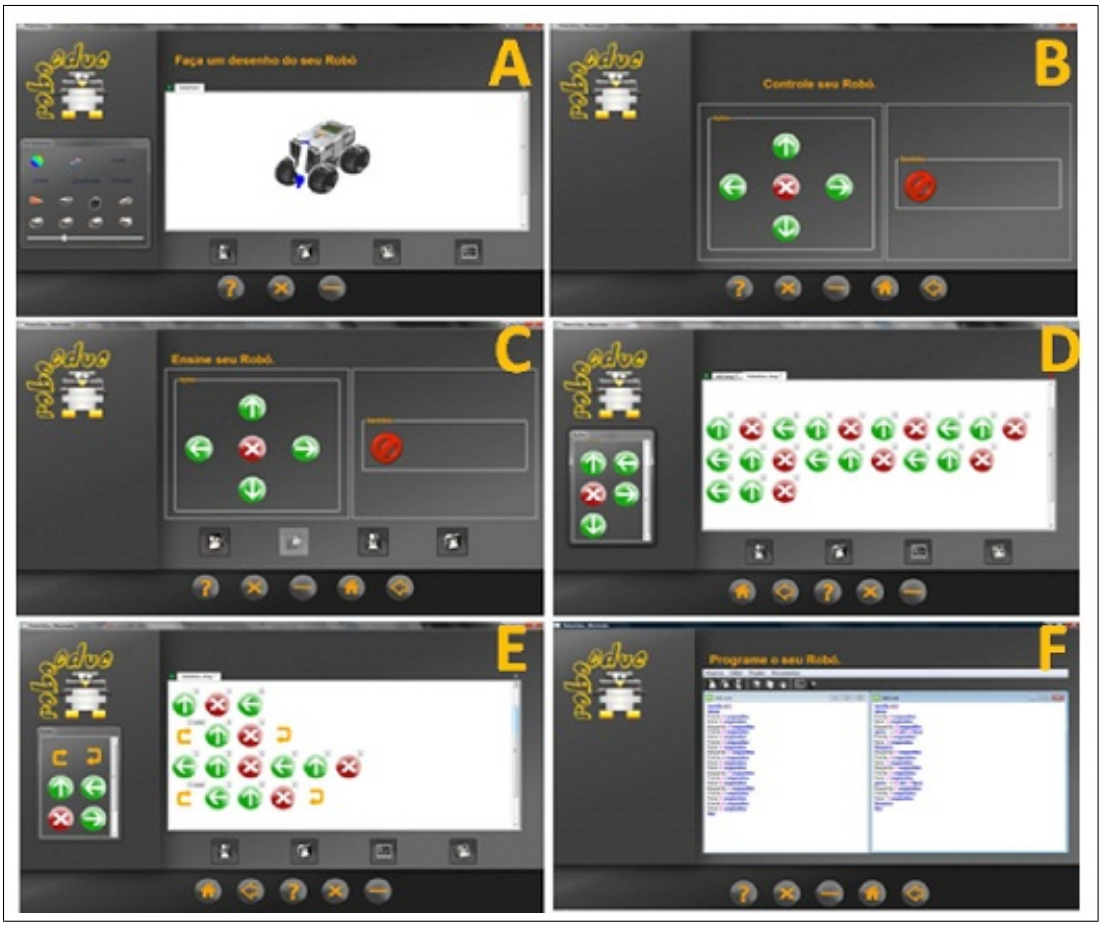

Figura 4.44: Módulos do **RoboEduc** - A) projetar, B) controle, C) ensinar, D) programação nível 1, E) programação nível 2 com estrutura de fluxo e F) programação textual nível 3.

Considerações:

Validamos o **RoboEduc 4.0** em oficinas de robótica ministradas em escolas públicas da periferia de Natal. Os alunos participantes das oficinas foram divididos em três grupos

com base em seus níveis de cognição, classificados em migrantes, nativos e analfabetos digitais.

Acreditamos que a estrutura da oficina foi relevante para o sucesso e desafios surgidos. Partimos de um processo exploratório, inicialmente com o projeto do robô a ser utilizado, depois para o concreto com a montagem com o kit LEGO, e por fim o processo de programação. É certo que os alunos dos grupos iniciais ainda fazem programações simples, por meio dos dois módulos usados (Controlar e Ensinar), porém isso não desqualifica o trabalho, pelo contrário, mostra que os alunos saíram de um processo de montagem para o de elaboração mental por meio das primeiras programações de seus protótipos. Os alunos do grupo classificado como nativos digitais utilizaram os demais módulos de programação com sucesso.

Consideramos também que contribuímos para o processo de letramento digital desses alunos envolvidos na oficina, pois proporcionamos a aquisição de habilidades básicas para o uso de Computadores e da Robótica, assim como contribuímos com o senso de cidadania e cultura.

4.5.2 Ambiente de simulação

Considerando a inviabilidade de alguns kits robóticos, foi necessário pesquisar outro tipo de ambiente que permitisse que um aluno aprenda a programação de robôs. Para isso, podem ser usados simuladores robóticos. Simuladores são ambientes virtuais que simulam um sistema real. No caso em questão, os simuladores podem ser utilizados para suprir a falta de robôs ou para poupar o uso de tal, caso testes possam gerar danos.

O ambiente de simulação apresentado em [da Costa Fernandes 2010] foi desenvolvido com o intuito de permitir que os alunos que não tem acesso a um kit de robótica possam aprender a programar robôs.

Para o desenvolvimento do simulador **RoboEduc** foi usada a linguagem de programação C++, juntamente com a plataforma Qt 4.6. Foram usados inúmeros recursos do Qt, procurando gerar uma interface amigável. Dentre esses recursos, merece destaque o *Graphic View Framework*. O Simulador é composto por 3 partes: a área de simulação, a área de botões e a lista de ações. A Figura 4.45 mostra uma imagem da tela do Simulador **RoboEduc**, destacando cada uma dessas áreas.

- Área de simulação
  Essa área é composta por objetos das classes do *Graphic View Framework*. A paisagem representa toda a área na qual o robô pode andar. A cena, embora não seja visível pelo usuário, possui todos os itens adicionados. Esses itens incluem o robô e os obstáculos (que podem ser quantos o usuário quiser).
- Área de botões
  A área de botões é composta por 6 botões: Adicionar obstáculos, Mostrar/Deletar caminho, e 4 botões representados pelos botões do Lego *Mindstorms* NXT que está na tela. As funcionalidades de cada um desses botões podem ser vista em [da Costa Fernandes 2010].
- Lista de ações

Figura 4.45: Tela principal do simualdor

A lista de ações mostra cada linha de código do programa que foi selecionado para ser executado. Quando o programa for alterado, os campos desta lista também irão alterar. Durante a execução do programa, o comando que está sendo realizado naquele momento fica em destaque. Dessa forma, o usuário pode analisar os comandos que o robô está realizando, conferindo se o resultado era o desejado.

A Figura 4.46 ilustra a mesma atividade realizada no experimento. No momento do experimento, o módulo simulador não estava finalizado, por este motivo não foi testado com os alunos.

Para ilustrar as suas funcionalidades foram feitos testes em laboratório simulando a mesma atividade do experimento.

4.6 Evolução das funcionalidades de RoboEduc

Nesta seção monstramos a Tabela 4.2. Uma tabela comparativa da evolução das versões do *software* **RoboEduc**. Com o passar dos testes e através das experiências vividas pelo grupo de desenvolvimento, as versões do *software* foram adquirindo novas funcionalidades, fazendo com que o *software* fosse evoluindo.

Figura 4.46: Atividade na qual o desafio era viajar no mapa do RN, passando por Natal, Caicó e Mossoró

Funcionalidades	Primeira	Segunda	Terceira	Quarta
Enviar Firmware	*	*	*	
Controlar Robô	*	*	*	*
Enviar Comando	*	*	*	*
Ensinar Robô			*	*
Executar Programa Aprendido			*	*
Programar Robô			*	*
Enviar Programa			*	*
Salvar Comando em Arquivo			*	*
Módulo Autoria			*	*
Projetar Robô				*

Tabela 4.2: Evolução das funcionalidades do **RoboEduc**

Capítulo 5

Considerações Finais

Como o processo de transição entre a sociedade industrial e a sociedade da informação gerou gradações de acesso à tecnologia. O desafio deste trabalho foi fornecer um *software* ao docente para o auxílio na educação digital contemplando os três níveis da sociedade atual que consistem em: analfabetos digitais, migrantes e nativos digitais, ao mesmo tempo. Assim será possível que os discentes, com diferentes níveis de cognição, possam realizar a mesma atividade proposta pelo docente respeitando as suas limitações e contemplando as suas competências.

A necessidade de minimizar o quadro de analfabetos digitais motivou o desenvolvimento deste trabalho. O acesso à tecnologia e a possibilidade de se ensinar programação para séries dos ensino infantil ao ensino superior através de diferentes níveis de programação gradativamente mais complexos foi realizado para favorecer o amadurecimento mental do aluno, no que diz respeito ao raciocínio-lógico exigido ao programar robôs. Os alunos-usuários da linguagem **Educ** partem de um sistema simples de programação para um mais complexo, obedecendo níveis de amadurecimento cognitivo conquistados com a prática, recorrentes do uso do *software* **Robo**.

Os resultados preliminares através dos testes de usabilidade mostraram que é possível fornecer um *software* capaz de auxiliar o docente a ministrar os conteúdos curriculares do ensino tradicional, mesmo tento alunos com diferentes níveis de aprendizado.

As contribuições deste trabalho são:

1. Análise de alguns *software* educacionais;
2. Análise das versões do *software* **RoboEduc**;
3. A versão 4.0 do *software* **RoboEduc**; Que tem como novas funcionalidades o projetar e o simulador 2D da linguagem **Educ**;
4. Validação da linguagem **Educ** para que possa auxiliar na educação digital contemplando as três níveis da sociedade atual ao mesmo tempo. Assim será possível que os discentes, com diferentes níveis de cognição, possam realizar a mesma atividade proposta pelo docente respeitando as suas limitações e contemplando as suas competências;
5. A migração da linguagem de baixo nível para NXC;
6. A migração do protocolo de comunicação para o *Bluetooth*;
7. Aprimoramento da linguagem **Educ** para suportar a interação com sensores de diversos tipos.

Encontramos também algumas desvantagens como o esforço de recrutamento do público-alvo do projeto, e da configuração do ambiente de teste que exige dedicação de uma equipe e tempo de preparação e realização dos testes (quatro ou no máximo 5 usuários por dia em testes de 1 hora de duração).

5.1 Publicações

Para consolidar o presente trabalho, algumas publicações em conferências e simpósios foram realizadas.

- SILVA, Akynara Aglaé Rodrigues Santos da ; Tomaz, Sarah ; AZEVEDO, Samuel ; Fernandes, Carla ; Barros, Renata Pitta ; Burlamaqui, Aquiles M. F. ; GONÇALVES, Luiz M G . O Aprender Brincando: a Robótica Pedagógica e suas Contribuições para o Ensino Infantil. In: XI International Symposium on Computers in Education, 2009, Coimbra. SIIE, 2009, 2009.
- Tomaz, Sarah ; SILVA, Akynara Aglaé Rodrigues Santos da ; AZEVEDO, Samuel ; Fernandes, Carla ; Barros, Renta Pitta ; Burlamaqui, Aquiles M. F. ; Silva, Alzira ; GONÇALVES, Luiz M G . ROBOEDUC: A PEDAGOGICAL TOOL TO SUPPORT EDUCATIONAL ROBOTICS. In: Frontiers in Education, 2009, San Antonio, Texas. Proceedings of the Frontiers in Education, 2009., 2009.
- Tomaz, Sarah ; Fernandes, Carla ; Barros, Renta Pitta ; SILVA, Akynara Aglaé Rodrigues Santos da ; AZEVEDO, Samuel ; GONCALVES, L. ; Burlamaqui, Aquiles M. F. . Integração da Robótica Educacional na Formação de Professores do Ensino Infantil. In: Workshop de Robótica Educacional - WRE, 2010, São Paulo. WRE 2010
- Freitas, Anelisa ; AZEVEDO, Samuel ; Barros, Renta Pitta ; SILVA, Akynara Aglaé Rodrigues Santos da ; Fernandes, Carla ; Leite, Luiz Eduardo Cunha ; Burlamaqui, Aquiles M. F. ; GONCALVES, L. . Uso de Robótica para Programas Educativos na TVDI. In: Workshop de Robótica Educacional - WRE, 2010, São Paulo. WRE 2010, 2010.
- Souto, Gustavo ; Fernandes, Carla ; Joany, Tassia ; NOBREGA, F. ; AZEVEDO, Samuel ; Burlamaqui, Aquiles M. F. . Ambiente de Simulação para Robótica Educacional. In: Simpósio Brasileiro de Informática na Educação - SBIE, 2010, João Pessoa. SBIE 2010, 2010.

Referências Bibliográficas

Aranibar, Dennis Barrios, Luiz M. G. Gonçalvez, Aquiles Burlamaqui, Marcela Santos, Gianna R. Araújo, Válber C. Roza, Rafaella A. Nascimento & Viviane Gurgel (2006), 'Technological inclusion using robots', *em Anais do II ENRI - Encontro nacional de Robótica Inteligente. Campo Grande, MS, Brasil* .

Aurélio (2010), 'Dicionário aurélio', `http://www.dicionariodoaurelio.com/`. Página da Internet - Último Acesso: 27/04/2010.

Barros, Renata Pitta (2008), 'Roboeduc - uma ferramenta para programação de robôs lego'. Trabalho de conclusão de curso (Engenharia da Computação). Universidade Federal do Rio Grande do Norte. Natal, RN.

Beetle, Elenco (2010), 'Elenco beetle', `http://www.robotshop.ca/elenco-beetle-robot-kit-1.html`. Página da Internet - Último Acesso: 30/04/2010.

brickOS (2010), 'Brickos operating system and c/c++ development environment for the lego mindstorms rcx controller', `http://brickos.sourceforge.net/documents.htm`. Página da Internet - Último Acesso: 30/04/2010.

Carvalho, Ana Amélia Amorim (2002), 'Teste de usabilidade: exigência supérflua ou necessidade?', *Actas do 5º Congresso da Sociedade Portuguesa das Ciências da Educação. Lisboa: Sociedade Portuguesa de Ciências da Educação* pp. 235–242.

Chella, Marco Túlio (2002), 'Ambiente de robótica educacional com logo', *XXII Congresso da Sociedade Brasileira de Computação - SBC2002. Florianópolis* .

Corporation, National Instruments (2010), 'National instruments', `http://www.ni.com/labview/whatis/`. Página da Internet - Último Acesso: 15/04/2010.

Curumim, XBot (2010), 'Xbot - robótica móvel inteligente para educação, pesquisa e entretenimento - robô educacional', `http://www.xbot.com.br/downloads/XBot_Curumim.pdf`. Página da Internet - Último Acesso: 25/04/2010.

da Costa Fernandes, Carla (2010), 'Ambiente simulado da metodologia roboeduc'. Trabalho de conclusão de curso (Engenharia da Computação). Universidade Federal do Rio Grande do Norte. Natal, RN.

da Educação do Brasil, Ministério (n.d.), *Guia das tecnologias do MEC*, Ministério da Educação do Brasil, http://portal.mec.gov.br/dmdocuments/guia$_t$*ecnologias*$_a$*tual*.*pdf*.

da Silva, Akynara Aglaé R. S., Maria das Graças P. Coelho, Renata Pitta Barros & Luiz Marcos G. Gonçalves (2008), 'A robótica pedagógica no contexto da educação infantil: Auxiliando o alfabetismo', *I Congresso de Tecnologias na Educação - 27 de outubro e 1^o de novembro de 2008.* p. 9.

da Silva, Alzira Ferreira (2009), RoboEduc: Uma Metodologia de Aprendizado com Robótica Educacional, Tese de doutorado, Universidade Federal do Rio Grande do Norte, Natal, RN.

de Abreu, Sueli, Márcia de Araújo Calçada & Julie Vinhaes (2002), *Manual do Usuário*, CNOTINFOR BRASIL Educação e Tecnologia Ltda, Minas Gerais, Brasil.

de Educação Tecnológica da PETe, Projeto (2008), *Manual do Usuário Módulo 2*, PNCA Robótca e Eletrônica LTDA, São Carlos, Brasil.

de Informática-UFRN, Superintendência (2010), 'Sigaa- sistema integrado de gestão de atividades acadêmicas', `http://www.sigaa.ufrn.br`. Página da Internet - Último Acesso: 30/04/2010.

Ferreira, Kátia Gomes (2002), 'Teste de usabilidade'. Trabalho de conclusão de curso. Universidade Federal de Minas Gerais. Belo Horizonte, MG.

Foundation, LOGO (2010), 'Logo foundation', `http://el.media.mit.edu/logo-foundation/`. Página da Internet - Último Acesso: 15/02/2010.

Group, The LEGO (2010), 'Lego mindstorms', `http://mindstorms.lego.com/en-us/default.aspx`. Página da Internet - Último Acesso: 24/04/2010.

Jeffrey, RUBIN (1994), 'Handbook of usability testing: How to plan, design and conduct effective tests.', *New York John Wiley Sons* p. 330.

LeJOS (2010), 'Lejos, java for the rcx', `http://lejos.sourceforge.net/`. Página da Internet - Último Acesso: 20/04/2010.

Malan, David J. & Henry H. Leitner (2007), 'Scratch for budding computer scientists', *Division of Engineering and Applied Sciences,Harvard University Cambridge,Massachusetts, USA* .

Nielsen, Jakob (1993), *Usability Engineering*, Morgan Kaufmann, San Francisco.

NQC (2010), 'Not quite c language with a c-like syntax for the lego mindstorms rcx controller', `http://bricxcc.sourceforge.net/nqc/`. Página da Internet - Último Acesso: 30/04/2010.

NXC (2011), 'Nxc$_g$*uide*',http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf. Página da Internet - Último Acesso: 30/01/2011.

PNCA (2010), 'A pnca robótica e eletrônica', http://www.pnca.com.br/. Página da Internet - Último Acesso: 26/04/2010.

Preece, Jenny, David Benyon, Gordon Davies & Laurie Keller (1993), *A Guide to Usability: Human Factors in Computing*, Addison Wesley Publishing Company.

RoboEduc (2010), 'Roboeduc - robótica educacional', http://www.roboeduc.com. Página da Internet - Último Acesso: 30/04/2010.

Santos, Marcela, Dennis Barrios Aranibar, Viviane Gurgel, Gianna R. Araújo, Válber C. Roza, Rafaella A. Nascimento & Luiz M. G. Gonçalves (2006), 'Roboeduc: A software for teaching robotics to technological excluded children using lego prototypes', *3rd IEEE Latin American Robotics Symposium - Universidade do Chile* .

Sommerville, Ian (2003), *Software Engineering*, 6ª edição, Pearson Education, São Paulo, Brasil.

Vallejo, Antonio Pantoja & Marlene Zwierewicz (2007), *Sociedade da Informação, Educação Digital e Inclusão*, CRC Press, Florianópolis, Brasil.

Vex (2010), 'Index tecnologia em robótica - vex', http://www.vexrobotics.com.br/index.php?id=2. Página da Internet - Último Acesso: 06/04/2010.

Wang, Eric & Ryan Wang (2001), 'Using legos and robolab (labview) with elementary school children', *31st ASEE/IEEE Frontiers in Education Conference - Reno, NV* .

Wang, Ting-Chung, Wen-Hui Mei, Shu-Ling Lin, Sheng-Kuang Chiu & Janet Mei-Chuen Lin (2009), 'Teaching programming concepts to high school students with alice', *ASEE/IEEE Frontiers in Education Conference - San Antonio, TX* .

Apêndice A

Plano de Teste RoboEduc 4.0

Software educacional para controle de dispositivos robóticos **RoboEduc 4.0**

A.1 Propósito do teste

O propósito deste teste é verificar o desempenho alcançado pelos professores/ monitores. Verificar o entendimento das funções do *software* utilizando o protótipo com a finalidade de realizar as alterações necessárias. Validar se é possível os discentes com diferentes níveis de cognição, possam realizar a mesma atividade proposta pelo docente. Será medido o tempo gasto para a realização das tarefas e serão identificados erros e dificuldades envolvendo a utilização do protótipo em tarefas rotineiras.

A.2 Declaração dos problemas

1. Os termos utilizados nas interfaces são intuitivos?
2. A ajuda *on-line* é eficaz?
3. O desempenho alcançado pelos usuários é o ideal?

A.3 Perfil do usuário

Serão utilizados sete usuários, um por dia. Os usuários devem ter de 17 a 25 anos de idade, nível médio (completo ou não) ou superior (completo ou não), mais de um ano de conhecimentos básicos de informática (uso do mouse e teclado) e de utilização de aplicativos básicos (como por exemplo, o LOGO), e não necessitam possuir conhecimentos técnicos em Robótica Educativa.

A.4 Metodologia

O teste será realizado com a finalidade de garantir a usabilidade do produto e será composto das seguintes partes:

1. Cada usuário será devidamente cumprimentado pelo avaliador, será orientado a se sentar e tentar se sentir confortável e relaxado. O usuário será orientado a preencher um pequeno questionário para identificação de seu perfil.
2. O usuário receberá um *script* introdutório de orientação do teste explicando o propósito e objetivos do teste, reforçando que o anonimato do produto deve ser mantido após os testes e o que é esperado dos usuários. Deve ser reforçado que o produto é o centro da avaliação e não o usuário e que as tarefas devem ser executadas de forma bastante confortável. Deve-se informar ao usuário que ele será observado e que estará sendo filmado.
3. Depois de passadas as orientações, será permitido que o usuário utilize o *software* livremente por cinco minutos. Logo depois, será requisitado ao usuário retornar à área de trabalho do *Windows* e lhe será entregue a lista de tarefas. O avaliador irá requisitar que o usuário verbalize suas dúvidas, pois isto ajudará ao avaliador anotar a ocorrência e a razão de problemas. Durante o teste, os acontecimentos observados pelo avaliador serão registrados em formulário próprio. Será cronometrado e registrado o tempo gasto na realização das tarefas.
4. Depois de completadas todas as tarefas, o usuário preencherá um questionário de avaliação do *software*, cuja finalidade é coletar informações preferenciais do usuário.
5. Logo após a etapa quatro, será dado uma pausa de dez minutos para o café.
6. O usuário será questionado pelo avaliador em uma sessão de questionamento. Serão discutidas percepções subjetivas de usabilidade do usuário acerca do *software*, realizados comentários globais sobre o desempenho do usuário e problemas encontrados. O usuário poderá comentar sobre o teste abertamente, permitindo uma coleta de informações complementares.
7. Depois da sessão de questionamento do usuário, será agradecido à colaboração do usuário e lhe será dado um brinde. Observação: o avaliador estará de posse do Roteiro do Avaliador para orientá-lo na condução do teste.

A.5 Lista de tarefas

Segue uma lista de tarefas preliminares para o teste de usabilidade do *software*.

Número da Tarefa	Descrição da Tarefa	Detalhamento da Tarefa REQ: Requerimentos para execução da tarefa; PR: Passos a serem realizados; TME: Tempo máximo para execução
1	Iniciar o *software* **RoboEduc**	REQ: O Computador deverá estar ligado e posicionado no Windows. A área de Trabalho do Windows deverá estar sendo visualizada. PR: O usuário aciona o Iniciar-> Programas-> **RoboEduc**. TME: 1,0 minuto
2	Você tem a necessidade de cadastrar algumas partes do Robô que brevemente será utilizada em sala de aula e farão parte do banco de dados do *software*. Realize o cadastro das seguintes partes: Modelo do Robô Sumô assim como a sua Base, Atuador, Sensores, Ações, Sentidos. As demais informações das partes devem ser completadas da maneira que você achar melhor.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela principal. PR: O usuário seleciona o módulo Professor pelo ícone à direita da tela. Na barra de *menus* no canto inferior da tela aciona o botão novo e digita as informações de cada componente a ser cadastrado e pressiona o botão confirmar. Sempre que for necessário cadastrar uma nova parte, repita o mesmo procedimento. TME: 5,0 minutos
3	Você deve cadastrar um robô com todos os componentes já criados para uma aula de sumô.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela do módulo Professor. PR: O usuário seleciona o modelo correspondente ao robô clicando no nome embaixo da figura que o representa. Seleciona as demais partes da mesma forma. Para navegar entre as abas utilize as setas de navegação que estão no canto superior direito da tela. Ao final, clique no botão confirmar. TME: 3,0 minutos
4	Você deve ensinar aos seus alunos como montar o robô sumô que você acabou de cadastrar.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela do módulo Aluno na opção Montar. PR: O usuário seleciona o modelo correspondente ao robô clicando no nome embaixo da figura que o representa. Seleciona as demais partes da mesma forma. Para navegar entre as abas utilize as setas de navegação que estão no canto superior direto da tela. Ao final, clique no botão confirmar. TME: 2,0 minutos

Tabela A.1: Lista de tarefas

Número da Tarefa	Descrição da Tarefa	Detalhamento da Tarefa REQ: Requerimentos para execução da tarefa; PR: Passos a serem realizados; TME: Tempo máximo para execução
5	Você não concluiu a sua aula passada, então nesta aula você deve ensinar ao aluno a buscar pelo robô que foi mostrado na aula passada, através do banco de dados do *software*.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela do módulo Aluno na opção Programar. PR: O usuário escolhe qual protótipo deseja utilizar e clica no botão confirmar. TME: 3,0 minutos
6	Para uma próxima aula de robótica, você deve consultar quais os robôs que já estão cadastrados no banco de dados, para saber se há ou não necessidade de cadastrar um novo protótipo.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela do módulo Professor. PR: O usuário, na barra de *menus* no canto inferior da tela aciona o botão Banco de Dados. Para navegar entre as abas utilize as setas de navegação que estão no canto superior direto da tela. TME:1,0 minuto
7	Você encontrou um erro no protótipo cadastrado e precisa deletar.	REQ: O *software* **RoboEduc** deverá estar sendo apresentado e posicionado na tela do Banco de Dados. PR: O usuário escolhe qual protótipo deseja deletar e na barra de *menus* no canto inferior da tela aciona o botão Deletar Protótipo. TME:1,0 minuto.

Tabela A.2: Continuação da lista de tarefas

Apêndice B

Questionário de avaliação do *software* pelo usuário

O objetivo deste questionário é colher informações sobre a opinião do usuário do teste de usabilidade que foi realizado utilizando o protótipo do *software* educacional para controle de dispositivos robóticos **RoboEduc 4.0**. As informações fornecidas são vitais para o aprimoramento do *software*. Nas questões de marcar, favor circular o número correspondente ao grau de concordância. Deverá ser marcada somente uma resposta por questão. Por favor, leia com atenção as questões a seguir, e em caso de dúvida, solicite esclarecimento com o avaliador.

1. Favor marcar o número correspondente ao grau que você concorda:

a. Facilidade de utilização	Dificil					Fácil
	0	1	2	3	4	5
b. Organização das informações	Ruim					Boa
	0	1	2	3	4	5
c. Layout das Telas	Confuso					Claro
	0	1	2	3	4	5
d. Nomenclatura utilizada nas telas (nome de comandos, titulos, campos, etc.)	Confuso					Claro
	0	1	2	3	4	5
e. Mensagens do sistema	Confusas					Claras
	0	1	2	3	4	5
f. Assimilação das informações	Dificil					Fácil
	0	1	2	3	4	5
g. No geral, a realização do teste foi	Monótona					Interessante
	0	1	2	3	4	5
h. O sistema é fácil de navegar, ou de trocar as janelas?	Confuso					Claro
	0	1	2	3	4	5
i. O sistema apresentou erros durante a execução?	Não					Sim
	0	1	2	3	4	5
j. Conseguiu realizar a tarefa dentro do tempo estipulado?	Não					Sim
	0	1	2	3	4	5

2. Aponte situações em que você achou fácil utilizar no *software*:

   _____

   _____

   _____

3. Aponte situações que você sentiu dificuldades:
   _____
   _____
   _____

4. Você utilizou a Ajuda *online* do *software* em algum momento? a. Sim b. Não Em caso afirmativo, descreva em quais situações você utilizou a Ajuda *online* do *software* (comente também se as informações da Ajuda *online* foram de pouca ou grande valia):
   _____
   _____
   _____

5. Diante do teste realizado, você acha que o *software* atingiu o objetivo para o qual foi desenvolvido? Explique.
   _____
   _____
   _____

6. O espaço abaixo é reservado para que você exponha sua opinião e sugira melhorias no *software*.
   _____
   _____
   _____

Apêndice C

Script de orientação

Olá, o meu nome é Renata Pitta, sou representante da **RoboEduc** e iremos trabalhar juntos nesta sessão de teste.

Estaremos efetuando o teste do protótipo de um produto destinado ao apoio das aulas de Robótica Educacional para uma escola de Robótica, cujo nome é **RoboEduc**.

O teste ocorrerá na sala em que estamos. Esta sala simula uma sala de aula, onde você permanecerá sentado em uma mesa. Você usará um *notebook Pentium III - 900 MHz* com o *Windows* 7 e o **RoboEduc** 4.0 instalados, lápis, caneta e papéis. Utilize o produto de forma normal e tranquila, como se estivesse usando um outro aplicativo.

É importante que você diga o que está pensando durante a execução das tarefas. Você poderá fazer perguntas, mas eu não poderei respondê-las. Isto irá ocorrer porque nós necessitamos verificar como você irá trabalhar com o produto de forma independente. Faça o melhor e não se preocupe com os resultados. É o produto que está sendo avaliado e não você. O produto ainda é um protótipo e com certeza, necessitará de modificações e você estará contribuindo para detectarmos quais são as modificações necessárias.

Eu me sentarei próximo a você para tomar algumas notas.

Jamier, também membro da **RoboEduc**, estará cronometrando o tempo gasto na execução das tarefas.

Estaremos sendo filmados e observados durante o teste. Você irá também responder a alguns questionários. É importante que seja utilizada informações verdadeiras e sinceras no preenchimento dos mesmos.

O nosso objetivo é descobrir falhas e vantagens na utilização deste produto de acordo com a sua perspectiva, portanto necessitamos saber exatamente o que você pensa.

Sua integridade será totalmente preservada, pois a filmagem será utilizada apenas para posterior análise dos testes por pessoal autorizado.

Estimamos cerca de uma hora para a duração desta sessão de testes.

Você tem alguma pergunta?

Se não, utilize o *software* livremente durante cinco minutos e esteja à vontade para fazer perguntas neste momento.

Agradecemos por sua colaboração.

Apêndice D

Questionário para identificação do perfil do usuário

O objetivo deste questionário é colher informações sobre o perfil do usuário do teste de usabilidade a ser realizado utilizando o protótipo do *software* educacional para controle de dispositivos robóticos **RoboEduc 4.0**. As informações fornecidas são vitais para o aprimoramento do *software*. Nas questões de marcar, favor circular a letra correspondente à resposta. A não ser que esteja indicado, deverá ser marcada somente uma resposta por questão. Por favor, leia com atenção as questões a seguir e em caso de dúvida, solicite esclarecimento com o avaliador.

1. Informações Pessoais
   (a) Qual é a sua idade? ——— anos.
   (b) Sexo: —— M. masculino F. feminino.
2. Informações Educacionais
   (a) Qual é o seu grau de instrução?
   a. 2° grau incompleto
   b. 2° grau completo
   c. 3° grau incompleto
   d. 3° grau completo
   e. Pós Graduação
   (b) Escreva o nome do curso que está fazendo ou que completou de acordo com o grau assinalado acima :————————
3. Experiência Profissional
   (a) Qual é a sua profissão? ————————-
   (b) Há quanto tempo se encontra nesta profissão?
   a. Menos de 6 meses
   b. Entre 6 meses e 1 ano
   c. Entre 1 ano a 2 anos
   d. Entre 2 anos a 4 anos
   e. Mais de 4 anos
   (c) A quanto tempo pesquisa/trabalha com o ambiente de Robótica Educacional?

a. Menos de 6 meses
b. Entre 6 meses e 1 ano
c. Entre 1 ano a 2 anos
d. Entre 2 anos a 4 anos
e. Mais de 4 anos

(d) Quais destes *softwares* de robótica tem conhecimento? (Pode-se marcar mais de uma opção)
a. Scratch
b. Alice
c. Imagine
d. RoboLab/ NXT Program
e. SuperLogo

4. Experiência Computacional

(a) Há quanto tempo você utiliza computador?
a. Menos de 6 meses
b. Entre 6 meses e 1 ano
c. Entre 1 ano a 2 anos
d. Entre 2 anos a 4 anos
e. Mais de 4 anos

(b) Em que local você utiliza computador? (Pode-se marcar mais de uma opção)
a. Em casa
b. No trabalho
c. Na escola/ Universidade
d. Outros, favor especificar: ———————————

(c) Em média, quantas horas por semana você utiliza o computador?
a. Menos de 2 horas
b. Entre 2 a 5 horas
c. Entre 5 a 10 horas
d. Mais de 10 horas

(d) Quais ferramentas abaixo você utiliza em suas atividades diárias? (Pode-se marcar mais de uma opção)
a. Windows
b. Word
c. Firefox/ Internet Explorer / Chrome
d. Paint
e. Outros, favor especificar: ———————————

Apêndice E

Tópicos para questionamento

O objetivo deste questionário é sugerir alguns tópicos a serem discutidos em uma sessão de questionamentos do usuário após a realização do teste do protótipo do *software* educacional para controle de dispositivos robóticos **RoboEduc 4.0**. Discussões a serem levantas:

1. Você acha que outras funcionalidades são necessárias neste *software*? Quais?
   _____
   _____
   _____

2. Você se sentiu confuso em algum momento durante a realização dos testes? Em quais momentos?
   _____
   _____
   _____

3. Você recomendaria a aquisição deste *software* a alguma escola? Por quê?
   _____
   _____
   _____

Apêndice F

Lista de tarefas

Agora, você dará início aos testes. Abaixo, nós temos 7 tarefas que devem ser executadas por você utilizando o *software*. As tarefas devem ser executadas na ordem em que se encontram. Você deve ler em voz alta cada tarefa antes de executá- la. Lembre-se:

- Verbalize suas dúvidas, pois isto ajudará ao avaliador anotar a ocorrência e a razão de problemas.
- É o produto que está sendo avaliado e não você.

F.1 Lista de tarefas

Tarefa 1: Iniciar o *software* **RoboEduc**

Tarefa 2: Você tem a necessidade de cadastrar algumas partes do robô que brevemente serão utilizadas em sala de aula e farão parte do banco de dados do *software*. Realize o cadastro das seguintes partes: Modelo do robô sumô assim como a sua base, atuador, sensores, ações, sentidos. As demais informações das partes devem ser completadas da maneira que você achar melhor.

Tarefa 3: Você deve cadastrar um robô com todos os componentes já criados para uma aula de Sumô.

Tarefa 4: Você deve ensinar aos seus alunos como montar o robô Sumô que você acabou de cadastrar.

Tarefa 5: Você não concluiu a sua aula passada, então nesta aula você deve ensinar ao aluno a buscar pelo robô que foi mostrado na aula passada, através do banco de dados do *software*.

Tarefa 6: Para uma próxima aula de robótica, você deve consultar quais os robôs que já estão cadastrados no banco de dados, para saber se há ou não necessidade de cadastrar um novo protótipo.

Tarefa 7: Você encontrou um erro no protótipo cadastrado e precisa deletar.

Apêndice G

Relatório Final

***Software* educacional para controle de dispositivos robóticos RoboEduc 4.0**

G.1 Sumário

Esse documento tem por objetivo apresentar os resultados do teste de usabilidade do *software* **RoboEduc 4.0**, originados de análises realizadas após os testes de usabilidade do protótipo do produto. Primeiramente são apresentadas as medidas coletadas durante o teste e as respostas aos questionários propostos aos usuários. Em seguida, são analisados os possíveis problemas de usabilidade da interface utilizando os dados previamente expostos e de outras informações do teste.

G.2 Método

Os testes de usabilidade foram realizados no **RoboEduc**, Empresa de Robótica Pedagógica incubada na Universidade Federal do Rio Grande do Norte, entre os dias 13 a 25/01/2011. Foram utilizados 4 usuários com a idade variando entre 18 e 30 anos. Com os dados do questionário de perfil dos candidatos observamos que a escolaridade dos mesmos predomina no ensino superior completo ou não. Os mesmos também têm mais de um ano de conhecimentos básicos de informática (uso do mouse e teclado) e utilizam aplicativos básicos (como por exemplo, o Logo). Os usuários são monitores de robótica pedagógica da empresa referida e trabalham com *softwares* de robótica pedagógica a mais de seis meses.

Os usuários preencheram questionários acerca do seu perfil e sobre satisfação do produto. O avaliador preencheu, em cada sessão de teste, um formulário de coleta de dados, onde foram registrados dados sobre o desempenho do usuário, número de erros encontrados, sucesso das tarefas e detalhes observados durante a execução de cada tarefa.

G.3 Resultado

1. **Tempo de execução das tarefas**

O Gráfico G.1 mostra as medidas de tempo de execução das sete tarefas realizadas no teste de usabilidade pelo usuário.

	usuário 1	usuário 2	usuário 3	usuário 4
Tarefa 1	3	2	30	28
Tarefa 2	420	300	600	273
Tarefa 3	120	85	3	95
Tarefa 4	125	40	76	46
Tarefa 5	40	120	14	50
Tarefa 6	30	65	20	63
Tarefa 7	12	10	49	25

Figura G.1: Resposta dos usuários sobre a interface

O Gráfico G.2 mostra a média e o desvio padrão calculado por tarefa.

Figura G.2: Média e desvio padrão por tarefa. Tempo em segundos

Tabela G.1: Número de erros cometidos

Tarefas	1	2	3	4	5	6	7
Usuário 1	0	1	0	2	0	0	0
Usuário 2	0	1	0	0	0	0	0
Usuário 3	1	1	0	0	0	1	0
Usuário 4	0	2	0	1	0	0	0
Média	0,25	1,25	0	0,75	0	0,25	0
Desvio Padrão	0,5	0,5	0	0,95	0	0,5	0

2. **Números de Erros na Execução das tarefas**
   A Tabela G.1 mostra o número de erros das sete tarefas realizadas no teste de usabilidade. Esta tabela apresenta também o valor médio e o desvio padrão das medidas coletadas.
3. **Respostas ao questionário de avaliação do *software* pelo usuário**
   O Gráfico G.3 apresenta uma avaliação das respostas dos usuários do teste ao questionário de avaliação do *software* onde as respostas são oferecidas ao usuário em uma escala 0 a 5.

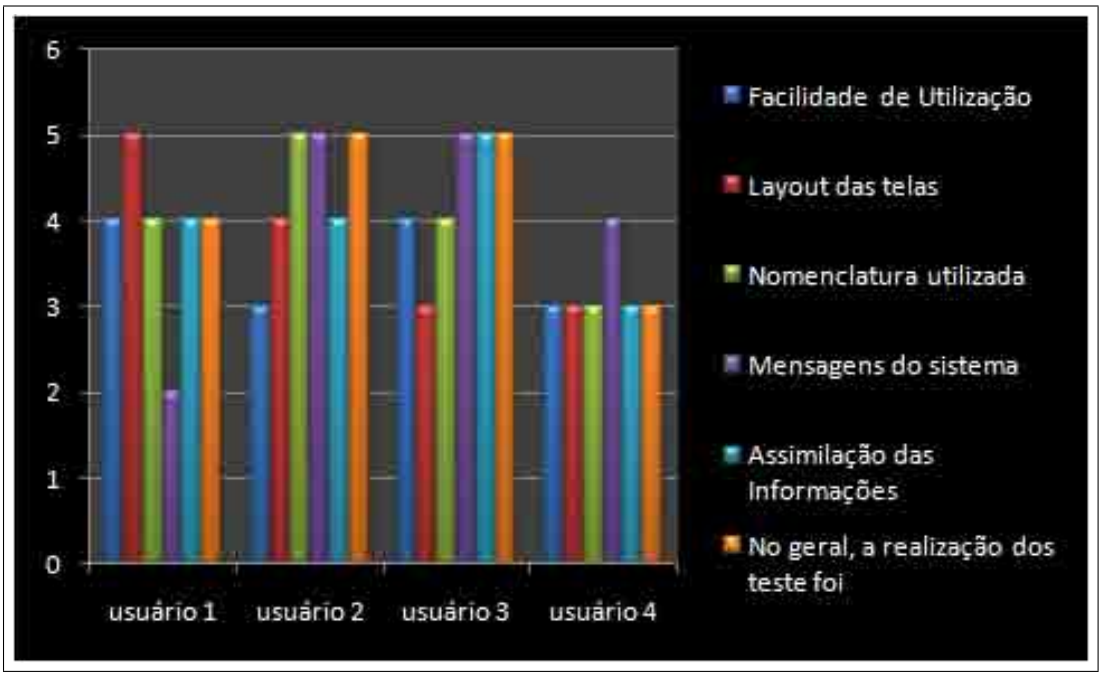

Figura G.3: Resposta dos usuários sobre a interface

Questão 1 - Aponte situações em que você achou fácil utilizar no *software*.
Usuário 1: " Iniciar o *software*, cadastrar as partes do robô e as descrições, bem como acessar o banco de dados."
Usuário 2: " Inicializar o *software* , adicionar ou remover tarefas e identificar tarefas através das imagens."
Usuário 3: " Depois de criar um componente ficou fácil criar os outros."

Usuário 4: " O *software* possui imagens que facilitam a sua utilização, como nos botões: confirmar, deletar e setas".

Questão 2 - Aponte situações em que você sentiu dificuldades.

Usuário 1: " Senti dificuldade na parte de cadastrar ações e construir o manual."

Usuário 2: " Nenhuma situação."

Usuário 3: " Achar a tela do banco de dados. Navegar entre as abas de modelo, base, atuador, sensor ação, sentido e descrição da tarefa através das setinhas de navegação."

Usuário 4: " Ao criar um novo modelo (protótipo) tive dificuldade em como fazê-lo."

Questão 3 - Você utilizou a Ajuda *online* do *software* em algum momento?

Usuário 1: " Sim. A Ajuda não foi muito explicativa: tentei voltar para tela principal e o botão ajuda não esclareceu. O botão não explica o que era cada coisa, ele apenas informa onde você deveria clicar (o passo a passo). Dessa forma o professor não sabe qual é o botão de montar, por exemplo."

Usuário 2: " Sim. Preencha os campos abaixo para criar uma nova base. Senti dificuldade de compreender a ordem em que as tarefas deveriam ser feitas."

Usuário 3: " Não. Na verdade só usei depois do teste para ver se as informações estavam organizadas."

Usuário 4: " Sim. Utilizei a ajuda para criar um novo protótipo, pois não estava conseguindo concluir a operação. A ajuda orientou no que estava faltando para que a tarefa fosse concluída."

Questão 4 - Diante do teste realizado, você acha que o programa atingiu o objetivo para o qual foi desenvolvido? Explique.

Usuário 1: " Sim, o teste é capaz de mostrar as facilidades, dificuldades e erros do programa."

Usuário 2: " Sim. O programa é bastante completo e apresenta-se como uma ferramenta facilitadora do ensino de programação em robótica educacional."

Usuário 3: " Sim, em linhas gerais ele funciona. A pessoa só precisa aprender suas funcionalidades."

Usuário 4: " Sim, o programa é capaz de criar modelos, controlar e programar robôs além de outras funcionalidades."

Questão 5 - O espaço abaixo é reservado para que você exponha sua opinião e sugira melhorias no *software*.

Usuário 1: " O *software* apresentou o *bug* na parte da descrição e o carregamento do banco de dados é relativamente lento. O botão ajuda poderia conter informações como: para cadastrar imagem clique em selecionar imagem, ou seja, explicar onde o professor deve clicar para realizar cada passo. A parte da ação poderia ser mais simples e os comandos mais autoexplicativos."

Usuário 2: " Melhorias: poder selecionar não apenas pela bolinha de seleção, mas também pela imagem na tela do montar protótipo. Maior rapidez para acessar o banco de dados."

Usuário 3: " O ícone do botão novo não é intuitivo. Na descrição não dá pra colocar um texto grande (só consegui 18 caracteres), depois de clicar no ícone manual apa-

recem mais caracteres e fica travando a digitação. Durante o cadastro do protótipo não dá para cancelar e voltar."
Usuário 4: " É um *software* eficaz e no geral de fácil utilização, o primeiro contato pode parecer difícil, mas ao utilizá-lo mais vezes observa-se suas diversas ações e como realizá-las."

G.4 Análise, discussão das descobertas e recomendações

A análise compreende a descrição dos problemas, uma possível proposta de solução e a prioridade para tal. A partir dos resultados obtidos no teste de usabilidade, foi realizada uma análise para detectar problemas de usabilidade e propor as respectivas alterações.

Análise 1: O problema de cadastrar as partes do robô

A Tarefa 2 foi a que teve o maior desvio padrão tanto na medida de tempo (S = 398,25) quanto na medida de número de erros (S = 1,25). Esse fato pode ser explicado de imediato pelo número de erros cometidos pelo usuário 4 consideravelmente maior que os outros três usuários, conforme mostra o Gráfico G.2 e a Tabela G.1. A Tarefa 2 consiste em cadastrar as partes que contêm o robô, cadastrando a base, os atuadores, os sensores, as ações e os sentidos necessários para montar o robô.

Durante a análise do teste, notou-se que os usuários demoraram a achar, na interface, onde cadastrar as partes a serem usadas para cadastrar um protótipo, especialmente na tela de ações e sentidos. As Tarefas 4 e 6 são semelhantes à Tarefa 5 e possuíram um desvio padrão médio. Este problema pode ser atribuído diretamente ao desenho confuso destinado ao banco de dados e pelo grande número de ícone que podem acessar esta interface.

Proposta de solução: inserir na interface de "nova ação" e "novo sentido" um botão "Exemplos". Quando este botão for acionado, será apresentada uma segunda tela onde poderão ser exibidos exemplos de como cadastrar uma nova ação ou um novo sentido. O campo informativo da ajuda *on-line* deve ser mantido para mostrar quais procedimentos devem ser seguidos para o preenchimento dos campos.

Prioridade: máxima.

Análise 2: Insuficiência da Ajuda *on-line*

A preocupação surgiu porque 3 usuários acessaram a Ajuda *on-line* básica. Atualmente, o protótipo possui Ajuda on-line disponível em algumas telas. Porém o texto não está adequado para suprir as necessidades dos usuários.

Proposta de solução: incluir ajuda mínima em todas as telas, com explicações diretas e resumidas de cada procedimento.

Prioridade: média.

Observações

O usuário 1 ficou confuso durante a execução de tarefa 2 e disse que não ficou claro como criar uma nova ação pois não entendia os comandos em inglês. O usuário 1 foi na tela de projetar robôs durante a execução da tarefa 4 ao invés de montar o Robô na tela de Aluno. Todos os usuários tiveram dificuldades em saber como criar um nova parte do robô para o cadastramento de um novo protótipo na tarefa 3. O usuário 3 concluiu que seria interessante mudar o estilo de acessar as abas das partes do robô, ficando livre

a escolha das abas. O usuário 3 realizou as tarefas 2 e 3 ao mesmo tempo, não ficando claro quando é apenas necessário o cadastro de um componente ou o cadastro do robô completo.

G.5 Conclusão

O teste de usabilidade do *software* **RoboEduc 4.0** mostrou sua utilidade na identificação dos problemas de usabilidade analisados. Com as técnicas de engenharia de usabilidade é possível não apenas identificar um problema, mas também o seu grau de importância, o impacto que pode causar frente aos usuários e o custo/benefício de sua manutenção. Concluiu-se que o *software* deverá sofrer alterações como: melhorar o tempo de acesso ao banco de dados, colocar a ajuda *on-line* em todas as telas, separar o cadastro de um componente e de um robô completo e outras alterações que melhorarão a interface ajudando a melhorar a usabilidade do produto.

G.6 Apêndice

Os formulários utilizados nas sessões de teste foram:

1. Roteiro do avaliador (Anexo A);
2. Questionário para identificação do perfil do usuário (Anexo D);
3. *Script* de orientação (Anexo C);
4. Lista de tarefas (Anexo F);
5. Questionário de avaliação do *software* pelo usuário (Anexo B);
6. Tópicos para questionamento (Anexo E).

Todos os formulários respondidos das sessões de teste se encontram em poder da avaliadora Renata Pitta Barros e poderão ser solicitados caso seja necessário.