UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEANDRO CAVALCANTE MENDONÇA LIMA

ESTUDO E TESTES DE USABILIDADE EM FERRAMENTAS DE AUTORIA VISUAL

São Luís
2012

LEANDRO CAVALCANTE MENDONÇA LIMA

ESTUDO E TESTES DE USABILIDADE EM FERRAMENTAS DE AUTORIA VISUAL

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Bel. Allan Kássio Beckman Soares da Cruz

São Luís
2012

Lima, Leandro Cavalcante Mendonça.

Estudo e testes de usabilidade em ferramentas de autoria visual/ Leandro Cavalcante Mendonça Lima. – São Luís, 2012.

95 f.

Impresso por computador (fotocópia).

Orientador: Allan Kássio Beckman Soares da Cruz.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2012.

1. Software – testes - usabilidade. 2. Software de autoria. I. Título.

CDU 004.42

Leandro Cavalcante Mendonça Lima

ESTUDO E TESTES DE USABILIDADE EM FERRAMENTAS DE AUTORIA VISUAL

Aprovada em:

31/08/2012

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Allan Kássio Beckman Soares da Cruz

Prof. Bel. Allan Kássio Beckman Soares da Cruz

(Orientador)

Mário Teixeira

Prof. Dr. Mário Antonio Meireles Teixeira

1º Examinador

Geraldo Braz

Prof. Me. Geraldo Braz Junior

2º Examinador

A todos aqueles que sempre acreditaram em mim e estiveram ao meu lado nos bons e maus momentos.

AGRADECIMENTOS

Ao meu falecido pai, Fernando Antonio Mendonça Lima, meu maior exemplo de paixão pelos estudos, de honestidade e de perseverança, minha inspiração para passar noites em claro fazendo trabalhos da universidade, para acordar antes das 6h da manhã e voltar para casa somente após as 22h todos os dias, para percorrer os quase 100km de trajeto diariamente, meu maior apoio nos momentos mais difíceis e a maior ajuda nas complicadas decisões relacionadas à carreira, sem os quais eu jamais chegaria nem perto de onde estou e do que sou atualmente. Obrigado por todo o tempo e investimento financeiro despendidos na minha educação e formação.

À minha mãe Maria dos Anjos Cavalcante Mendonça Lima, por todo o carinho, atenção e amor. E ao meu irmão, Lucas Matheus, pelo companheirismo.

Obrigado aos professores dos laboratórios que participei por dois anos, professores Denivaldo Cícero Pavão Lopes e Francisco José da Silva e Silva, respectivamente do LESERC e LSD, obrigado pelo primeiro contato com o mundo profissional, o conhecimento adquirido levarei comigo para sempre.

Aos colegas de trabalho da ALUMAR pelos dois anos de conselhos, paciência, ideias, incentivos e reconhecimento que sempre tiveram para comigo, nas mais diversas situações, obrigado por terem acreditado em mim.

Aos colegas de universidade, tanto da UFMA quanto do IFMA, os quais me fizeram companhia nas 14 cadeiras feitas em média por semestre. Suas presenças com certeza fizeram todas elas mais fáceis de serem concluídas. Aqui gostaria de deixar um obrigado em especial para os amigos mais próximos, Leandro Rodrigues, Igor Fernando, Bruno, Suellen, Luciano, Jhonatas, Almir e Thiago, amizades que, acredito, nunca acabarão.

Aos meus tios e tias, pelo suporte e ajuda sempre que necessários, aos meus primos pela companhia e discussões sobre a vida, aos meus avós que sempre me acolheram durante a correria da vida.

À minha namorada e amiga, Andrea Braga, pelo carinho, compreensão e apoio durante boa parte de toda essa jornada.

Aos amigos de infância, que mesmo seguindo caminhos distintos nunca deixaram de se preocupar e manter contato, Paulo, Ricardo e Ramon.

Obrigado ao professor Allan Kássio Beckman Soares da Cruz, sem os seus conselhos e ideias este trabalho não seria possível.

Obrigado a todos que fizeram parte da minha vida durante esses últimos quatro anos e meio. Tenho certeza que sem a companhia de todos não teria conseguido chegar até aqui.

“Você pode conquistar tudo que você quiser.”

Fernando Antônio M. Lima

RESUMO

Com a grande quantidade de softwares disponíveis no mercado, os testes dos mesmos são cada vez mais importantes para o sucesso de uma aplicação, devido ao aumento da exigência dos usuários em relação a questões anteriormente não tão importantes como, nesse caso, a qualidade das interações com a interface do sistema. Neste trabalho, são apresentados os diversos conceitos que envolvem interação humano-computador, usabilidade e softwares de autoria, dando ênfase aos testes de usabilidade, sua preparação, procedimentos, desenvolvimento e apresentação de resultados. Com o objetivo de possibilitar a preparação de um modelo padrão mais efetivo de interface e interações de softwares de autoria automatizada de conteúdo.

Palavras-chave: Usabilidade, Autoria, Softwares de Autoria, Teste de Usabilidade, IHC.

ABSTRACT

With the large amount of software available in the market, their tests are increasingly important for the application sucess, since users are increasingly demanding in relation to issues before not so important, as the quality of interaction with the system's interface. This work presents several concepts involving human-computer interaction, usability and authoring softwares, emphasizing usability testing, their preparation, procedures, development and presentation of results, in order to enable the preparation of a more effective standard model of interface and interactions of automated content authorship software.

Word-keys: Usability Test, Usability, Authorship, Authorship Software, IHC.

RESUMEN

Con la gran cantidad de software disponibles en el mercado, las pruebas de los mismos son cada vez mas importantes para el éxito de una aplicación, debido a que los usuarios están cada vez mas exigentes en relación a preguntas anteriormente no tan importantes, como la calidad de la de la interface del sistema. En este trabajo son presentados los diversos conceptos que envuelven la interacción humano-computador, usabilidad y software para desarrollo. Enfatizando las pruebas de usabilidad, su preparación, procedimientos, desarrollo y presentación de resultados. Con el objetivo de facilitar la preparación de un modelo mas efectivo de interface e interacciones de los software para desarrollo automatizada de contenido.

Palabras Clave: Usabilidad, Autoría, Software para Desarrollo, Pruebas de Usabilidad, IHC.

LISTA DE ABREVIATURAS OU SIGLAS

UFMA: Universidade Federal do Maranhão

PC: *Personal Computer* (Computador Pessoal)

TI: Tecnologia da Informação

TIC: Tecnologia da Informação e Comunicação

IHC: Interação Humano-computador

API: *Application Programming Interface* (Interface de programação de Aplicativos)

TVDi: TV Digital Interativa

WEB: World Wide Web

ISC: Illumination Software Creator

HTML5: Hypertext Markup Language, versão 5

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O Curso de Ciência da Computação da Universidade Federal do Maranhão (UFMA) na sua formação básica possui um sub-eixo formativo cujo objetivo visa o domínio dos fundamentos e das técnicas básicas da computação, o desenvolvimento do raciocínio lógico e da habilidade de resolução de problemas, da organização e manipulação de informações; da organização e arquitetura de computadores e da utilização de técnicas e ferramentas básicas (Teixeira et al., 2007). Esse sub-eixo proporciona ao aluno apreender o conceito de programa computacional. Um programa pode ser descrito como um conjunto estruturado de instruções que capacitam uma máquina aplicar sucessivamente certas operações básicas e testes em uma parte determinada dos dados iniciais fornecidos, até que esses dados tenham se transformado numa forma desejável (Menezes e Diverio, 2000). Apesar de parecer simples e objetivo, esse conceito abrange uma série infinita de combinações e de resultados finais. Por esse motivo é tão difícil criar um programa que supra as necessidades das pessoas, conhecidas como usuários, para as quais foram projetados. Mesmo seguindo padrões, modelos, normas, metodologias das mais diversas, criar programas, ou seja, programar é uma atividade abstrata (Castro, 2011).

O desenvolvimento de programas de computador, também conhecidos como aplicativos, é uma atividade que já existe há algum tempo. Turing (1936) foi o primeiro a descrever o que viriam a ser os programas utilizados hoje em dia, contudo, no início, essa era uma atividade restrita a um pequeno grupo de profissionais e isso acontecia por diversos motivos: inexistência de interfaces, difícil acesso à dispositivos capazes de processar instruções, conhecimento muito específico e novo etc. Entretanto, atualmente, o cenário é muito diferente. Devido à popularização dos computadores pessoais (PCs) e o surgimento e posterior massificação da internet, praticamente todas as atividades intelectuais são realizadas utilizando algum tipo de dispositivo computacional como

ferramenta de auxílio, seja um computador, um *tablet*, ou até mesmo um dispositivo embarcado (Barbosa e Da Silva, 2010).

Desse modo, muitas pessoas, mesmo aqueles que não são programadores ou estudiosos do ramo de Tecnologia da Informação (TI), têm sentido a necessidade de criar seus próprios aplicativos para facilitar o seu trabalho e até sua vida como um todo.

Direcionados para as necessidades desse público, surgiram os aplicativos chamados de programas de autoria ou softwares de autoria. Eles são empregados a fim de abstrair do autor toda, ou pelo menos parte da complexidade de se utilizar uma linguagem de programação na criação de aplicações (Guimarães, 2007). Ou seja, são basicamente programas que são utilizados como ferramentas para auxiliar no desenvolvimento de outros programas. O objetivo principal é criar maneiras alternativas, sem a necessidade da programação em si, para criar os aplicativos.

Este trabalho abrange duas visões que surgiram devido a diferentes necessidades. Uma delas é a dos usuários que precisam criar seus próprios softwares, jogos ou animações, e a outra é a dificuldade dos programadores em fornecer ferramentas que atendam esses anseios dos usuários, mas de forma simples, completa, efetiva e prática.

A motivação do mesmo surgiu em função da dificuldade de mensurar a qualidade das interações dos usuários com os diferentes softwares produzidos atualmente, pois a diversidade de softwares é muito grande. Por exemplo, apenas a Apple, na sua loja de aplicativos para dispositivos móveis, possui atualmente mais de 500.000 aplicativos em suas versões finais disponíveis para os seus usuários (Apple Inc., 2012), e como são objetos abstratos, não existe uma maneira objetiva e exata que consiga mensurar se a interatividade dos mesmos é adequada ou não. Existem diversas classes ou categorias de aplicativos. A classe escolhida para ser o objeto de estudo desse trabalho foi a

dos softwares de autoria, pois seu foco é justamente usuários que não têm grandes conhecimentos sobre desenvolvimento de aplicações.

O objetivo trabalho monográfico é realizar um estudo da interação dos usuários com diversos softwares de autoria por intermédio de testes de usabilidade. Estudou-se, desse modo, a interação dos usuários com os sistemas computacionais, as diferentes formas de aplicação dos testes de usabilidade, tal como identificar a mais efetiva para cada sistema de autoria. Apresenta-se, também, uma comparação entre os resultados dos testes e, por fim, expõem-se uma interface modelo para ser utilizada como padrão em softwares de autoria, baseada nos conhecimentos dos resultados obtidos nos testes.

O restante deste trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta a origem e os conceitos de Interação Humano-Computador (IHC) e de Usabilidade, seus benefícios, além de exemplos dos mesmos, às diferentes visões dos sistemas interativos e engenharia semiótica.

O Capítulo 3 mostra o que são softwares de autoria, sua origem, sua utilização e apresenta os três softwares que serão utilizados para a realização dos testes.

O Capítulo 4 aborda os testes de usabilidade, seus conceitos, metodologias, aplicações e resultados, além do conceito e aplicação de Personas.

O Capítulo 5 mostra a proposta do modelo resultante da análise dos resultados dos testes.

O Capítulo 6 apresenta a conclusão do trabalho, além de sugestões de trabalhos futuros.

1.1 Trabalhos relacionados:

A utilização de testes de usabilidade para identificar possibilidades de melhorias nas interações humano-computador já existe há um certo tempo. Entretanto, recentemente, as mesmas vêm ganhando uma importância cada vez maior na influência do desenvolvimento de softwares.

Em Mathew e Abri (2011) é realizado um estudo sobre o andamento do de IHC na atualidade, os novos desafios e avanços que têm surgido juntamente com as novas tecnologias e como seus principais conceitos se aplicam nessa nova realidade.

Porém, um dos primeiros estudiosos da área de IHC foi Nielsen (1993), que já nesse período defendia que não importa a *expertise* do desenvolvedor ou *designer,* o mesmo não conseguirá criar interfaces perfeitas em uma única tentativa. A aproximação do ideal ocorre apenas depois de um grande ciclo de engenharia de usabilidade e estudo das interações.

Uma das formas de estudo dessas interações são os testes de usabilidade que já em Rosenbaum e Walters (1988) são citados seus elementos e aplicação como forma de melhorar a qualidade das interações dos sistemas da Xerox.

Kantner (1994) expõe as diversas técnicas para realizar um teste de usabilidade, como planejar, preparar, conduzir, realizar e até mesmo como tirar conclusões dos resultados dos testes de usabilidade.

Os softwares de autoria tais quais os abordados nesse trabalho são estudados em Roddy et al. (1966), em que o desenvolvimento de aplicações com geração automatizada é comparado com a maneira tradicional de programação em busca do resultado final.

No entanto, atualmente, já existem softwares de autoria para as mais diversas plataformas e com diversos paradigmas. Lima et al. (2010), por exemplo, apresenta o Composer 3, um ambiente de autoria extensível, adaptável e multiplataforma que tem foco em aplicações para TV Digital Interativa (TVDi).

Portanto, para este trabalho, estudou-se as interações dos usuários com os softwares de autoria, através da utilização da avaliação de IHC chamada teste de usabilidade.

2 INTERAÇÃO HUMANO-COMPUTADOR E USABILIDADE

Figura 2.1 - Evolução da interface dos *joysticks* dos consoles da Nintendo

Uma interface é o meio físico ou lógico através do qual um ou mais dispositivos ou sistemas incompatíveis conseguem comunicar-se entre si (Ferreira, 2009). A Figura 2.1 acima é um exemplo da evolução das interfaces físicas. A interface é considerada a embalagem do software de computador, portanto se ela for fácil de aprender, simples de usar, direta e amigável, o usuário estará inclinado a fazer bom uso da mesma (Pressman, 1995) e, além disso, a grande maioria dos usuários acredita que o sistema é a interface com a qual entram em contato (Hix e Hartson, 1993). As áreas de IHC e de usabilidade se concentram no estudo dessas interfaces. A Figura 2.2 abaixo mostra um exemplo da evolução das interfaces nos sistemas operacionais.

Figura 2.2 - Evolução na interface do Sistema Operacional Windows da Microsoft

2.1 Interação Humano-Computador

O comportamento humano não é sempre igual, mas cheio de surpresas, o que dificulta o estabelecimento de simples verdades sobre o que esperar das pessoas em determinadas situações (Lindgaard, 1994). Esse foi um dos motivos que levaram as pessoas ao estudo de IHC.

A Interação Humano-Computador (IHC) é a área da computação que investiga o design, avalia e implementa interfaces para que seres humanos possam interagir com sistemas computacionais de maneira eficiente e intuitiva (Santos e Teixeira, 2010).

A interação dos seres humanos com os dispositivos computacionais acontece há muito tempo. O primeiro *mouse*, como mostrado na Figura 2.3 é exemplo de um dos primeiros dispositivos construídos para facilitar essa interação, foi construído em 1963 por Douglas Engelbart (Edwards, 2008). Mas, sem dúvida, essas interações evoluíram muito. Os *tablets* podem ser citados como uma dessas evoluções, ver Figura 2.4.

Figura 2.3 - O Mouse criado por Douglas Engelbart em 1963.

Figura 2.4 - IPad, o tablet da Apple lançado em 2010.

Sabe-se que os computadores pessoais atuais apenas conseguem computar a linguagem binária, então cada estímulo, cada interação com os mesmos não será nada além de uma forma de transmitir zeros e uns de modo que possam ser processados (Stallings, 2003). Uma das primeiras formas de interação foi o uso de cartões perfurados, em que os programas eram escritos em cartões e inseridos no mainframe para processamento. Com o tempo surgiram novas formas de interação: teclado e mouse; toque, através de telas sensíveis; voz, através de microfones que captam e reconhecem comandos em fala natural; imagem, através de câmeras e sensores que captam movimentos etc (Gadelha, 2009).

Além das tecnologias, os estudos de IHC também têm avançado. Atualmente já se fala em interações adaptáveis e inteligentes, ou seja, softwares que podem se modificar para atender aos requisitos de usabilidade da interação de cada usuário em cada período de tempo (Mathew e Abri, 2011). Um exemplo de interação adaptável são os sites de busca online, que conseguem personalizar os resultados para cada usuário de acordo com pesquisas realizadas anteriormente ou sites visitados recentemente.

Todas essas inovações e evoluções visam melhorar as formas de interações, facilitar as mesmas, tentar simular interações do mundo real através do uso de computadores e máquinas.

Esse é o campo de estudo de IHC, seus objetivos são aprimorar as interações entre usuários e computadores, aumentar a qualidade do uso dos softwares, diminuindo, ao mesmo tempo, o seu impacto na vida dos usuários. Quanto mais fácil e simples for a interação, maior será a usabilidade de determinado sistema, assim como a complexidade por trás do mesmo, ou seja, do seu desenvolvimento (Mathew e Abri, 2011).

Como os estudos em usabilidade e IHC são, de certa forma, recentes se comparados com o surgimento dos primeiros PCs, estas áreas do conhecimento ainda têm muitos desafios e o desenvolvedor deve ter consciência que o resultado do seu trabalho vai influenciar e modificar a vida de um grande número de pessoas das mais diversas formas.

2.1.1 Benefícios do estudo e utilização de IHC

Estudar as interações entre as pessoas e os computadores possibilita melhorar a percepção que temos sobre essas interações e assim conseguir criá-las, modificá-las e utilizá-las de maneira que facilitem a experiência de utilização dos usuários em geral. Com o aumento da experiência de uso, a qualidade da utilização dos sistemas também aumenta, contribuindo para diversas questões que influenciam no desenvolvimento de aplicativos.

Em primeiro lugar, pode-se dizer que aumenta a produtividade dos usuários, pois, se a interação for eficiente, os usuários podem receber apoio computacional para alcançar seus objetivos mais rapidamente. Outros fatores que tem um crescimento são as vendas e a fidelidade do cliente, pois os clientes satisfeitos recomendarão o sistema a seus colegas e amigos e também voltam a comprar novas versões (Barbosa e Da Silva, 2010).

Em se tratando de reduções, o número e a gravidade dos erros cometidos pelos usuários diminuem, pois eles poderão prever as consequências de suas ações e compreender melhor as respostas do sistema e as oportunidades de interação. Sendo assim, o custo de treinamento também cai, pois os usuários poderão aprender durante o próprio uso e terão melhores condições de se

sentirem mais seguros e motivados para explorar o sistema e, por último, o custo de suporte técnico também sofre reduções, pois os usuários terão menos dificuldades para utilizar o sistema e, se cometerem algum erro, o próprio sistema oferecerá apoio para se recuperarem dos erros cometidos (Barbosa e Silva, 2010).

Além de todos esses pontos, levar em consideração os princípios de IHC desde o início do planejamento e desenvolvimento de sistemas propicia uma economia de tempo e custos devido à menor quantidade de modificações e testes necessários nas fases finais dos mesmos. Bias e Mayhew (2005) apresentam estudos indicando retorno de investimentos em qualidade de uso e IHC. Outro fator positivo é a elevada satisfação dos clientes, usuários, contratantes etc, pois, um software com boa usabilidade tem muitas vezes a rotulação como sendo um software de qualidade. Fato que implica diretamente na fidelização dos clientes, e novamente em lucros.

2.2 Usabilidade

A usabilidade de um sistema está relacionada com a experiência que os usuários têm com o mesmo, sua interação com a interface (Ferreira, 2002). Deve ser um processo interdisciplinar e colaborativo (Botella et al., 2011). Sendo assim, é importante que todos os envolvidos no processo de criação das soluções tenham consciência dos seus princípios básicos e estejam atentos para o mesmo durante toda a fase de desenvolvimento.

Gibson (1977) definiu o termo *affordance,* na Psicologia, que mais tarde foi adaptado para IHC por Norman. *Affordance* de um objeto corresponde ao conjunto das características de um objeto capaz de revelar aos seus usuários as operações e manipulações que eles podem fazer com ele (Norman, 1988). Em uma interface gráfica, por exemplo, a *affordance* de um botão de comando diz

respeito à possibilidade de pressioná-lo usando o mouse ou o teclado e, assim, acionar uma operação do sistema (Barbosa e Silva, 2010). Podemos dizer que um sistema com boa usabilidade possui bons *affordances*.

Figura 2.5 - Exemplo de bons *Affordances*

Figura 2.6 - Exemplo de maus *Affordances*

Segundo a norma ISO/IEC 9126 (1991), que define critérios de qualidade de software, usabilidade é “um conjunto de atributos relacionados com o esforço necessário para o uso de um sistema interativo, e relacionado com a avaliação individual de tal uso, por um conjunto especifico de usuários”.

Já a norma ISO 9241-11 (1998), que trata de requisitos de ergonomia, define usabilidade como “o grau em que um produto é usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico”.

Essas duas definições são bem amplas, mas pode-se, contudo, já ter uma noção da abordagem orientada ao produto e ao usuário do mesmo, em comparação com a abordagem tradicional focada no desenvolvimento.

Para Nielsen (1993), o critério de usabilidade é um conjunto de fatores que qualificam quão bem uma pessoa pode interagir com um sistema interativo. Esses critérios estão relacionados com a facilidade e o esforço necessários para os usuários aprenderem e utilizarem um sistema. Desse modo, a usabilidade endereça principalmente a capacidade cognitiva, perceptiva e motora dos usuários empregada durante a interação. Os fatores de usabilidade por ele considerados são:

- Facilidade de aprendizado – está relacionado com o tempo e esforço que o usuário necessita para poder ter determinado nível de performance e desempenho no uso do sistema.
- Facilidade de recordação (no caso de uso intermitente) – está relacionado com quantidade de esforço que o usuário precisa ter para lembrar de determinada ação já previamente utilizada.
- Eficiência ou produtividade– está relacionada com a performance, ou seja, o tempo que o usuário necessita para realizar determinada tarefa com o uso do sistema.
- Segurança de uso – está relacionada com a eficácia do sistema em proteger o usuário de situações que possam prejudicá-lo acidentalmente.
- Satisfação do usuário – está relacionada com uma avaliação subjetiva dos usuários em relação à utilização do sistema.

Segundo Ferreira (2002), a usabilidade é relacionada à eficácia e eficiência da interface diante do usuário e a reação do usuário diante da interface.

Caso os critérios de usabilidade não sejam satisfeitos, há grandes chances de haver problemas de usabilidade. Eles ocorrem quando um usuário ou um grupo de usuários encontra dificuldades para realizar uma tarefa com uma interface. Tais dificuldades podem ter origens variadas e ocasionar perda de dados, diminuição da produtividade e podem chegar até mesmo à rejeição total do software por parte dos usuários (Ferreira, 2002).

Desenvolver sistemas com boa usabilidade significa focar o desenvolvimento nos usuários. Para tanto, deve-se observar suas características, seu contexto de utilização e o seu objetivo com a utilização do sistema.

O ideal seria que todos os fatores de usabilidade fossem amplamente aplicados em qualquer sistema, mas cabe ao Analista e Designer verificar esta possibilidade, visto que é bastante complicada a combinação de todos e, em alguns casos, até mesmo conflitante. Cada sistema deve ter os fatores que mais impactam no seu público alvo com maior importância, dando prioridades a cada fator. Segundo Perry (2003), existe a necessidade de entender o trabalho que extrapola o indivíduo. Entender o processamento de informação e a resolução de problemas possibilita incorporar o uso das ferramentas para obter a melhor usabilidade possível.

Desprender tanto esforço com usabilidade pode, a princípio, parecer um custo a mais no desenvolvimento, mas, na verdade, com o seu uso, grande parte dos benefícios de IHC serão alcançados, trazendo um retorno futuro. Por exemplo, um grupo de usuários pode encontrar dificuldades em utilizar um sistema, dificuldades em realizar alguma tarefa devido a uma interface que não é clara o suficiente para possibilitar tal interação. Provavelmente, esses usuários vão utilizar cada vez menos tal sistema, ou então irão solicitar grandes alterações no mesmo. Desse modo, a necessidade de suporte, manutenção e treinamentos aumentará bastante, tudo isso contribuindo para um aumento nos

custos que seriam muito menores, caso o foco em usabilidade existisse desde os princípios da concepção do sistema.

2.2.1 Diretrizes de usabilidade

Diretrizes de usabilidade são orientações para o desenvolvimento de sistemas tendo como foco o usuário.

As principais diretrizes são:

- Contexto e navegação: o usuário deve rapidamente compreender o que é e como funciona o sistema, localizar o que busca com facilidade e realizar suas tarefas sem dificuldades;
- Carga de informação: quanto menor a quantidade de informação que o usuário venha a ter acesso melhor, para que o mesmo consiga focar a atenção no seu objetivo, tarefa;
- Autonomia: o usuário deve conseguir autonomia na utilização do sistema, conseguindo executar a sua tarefa sem complicações.
- Erros: todo erro, indisponibilidade ou falha previstos devem ser esclarecidos para os usuários, e todo erro cometido pelos mesmos deve ser passível de correção;
- Desenho: o design do sistema deve seguir padrões, tem que ser claro, simples e ainda assim garantir que todas as informações necessárias estejam visualizáveis.
- Redação: o sistema deve utilizar linguagem amigável, com frases e conceitos familiares aos usuários, além de ser objetivo;
- Consistência e Familiaridade: o usuário deve se sentir confortável ao usar o sistema, para isso o mesmo deve ser adequado à experiência de vida do usuário (Brasil, 2010).

2.2.2 Engenharia de Usabilidade

Engenharia de usabilidade é um conjunto de atividades que devem ocorrer durante o desenvolvimento de um sistema para que o mesmo tenha uma boa usabilidade (Nielsen, 1993). Constituem essas atividades o conhecimento do usuário, que consiste em estar ciente de suas características pessoais, seu ambiente físico e social; a realização de uma análise competitiva, ou seja, examinar sistemas com funcionalidades semelhantes ou complementares do mercado, analisar seus erros e acertos, deficiências e qualidades; a definição de metas de usabilidade, que significa definir quais fatores que impactam na usabilidade terão maior prioridade no projeto; a adoção do design participativo, que consiste em ter acesso, durante toda a fase de desenvolvimento, a algum usuário representante de determinada classe de tipo de usuário; aplicar diretrizes e análise heurística, isto é, desenvolver baseando-se em diretrizes bem definidas, analisando regularmente se as mesmas têm sido cumpridas; e, por fim, a realização de testes empíricos, ou seja, fazer uma análise dos usuários durante a utilização do sistema.

Como se pode observar existem várias métricas de usabilidade, várias formas de aplicação, diversas metodologias. Os testes de usabilidade desse trabalho consistirão em uma análise levando em conta os principais pontos de cada técnica e abordagem.

2.3 Diferentes visões de sistemas interativos

Quando se fala em desenvolvimento de sistemas ou softwares, existem diversas classes de profissionais envolvidos e cada tipo de profissional tem uma visão diferente do mesmo produto. São muitas as questões correlacionadas a esse fato, elas podem ser de origem cultural, financeira, etc.

Um grande desafio nessa área é a identidade cultural de cada usuário e dos desenvolvedores. Essas questões influenciam diretamente em como cada um esta habituado a interagir com certa classe de aplicações, e no caso dos desenvolvedores, na forma de programar o aplicativo. Um exemplo claro disso são as diferenças de websites ocidentais para os orientais, diferenças que se devem a inúmeros fatores culturais. A própria forma de escrever da direita para esquerda pode ser citada nesse contexto.

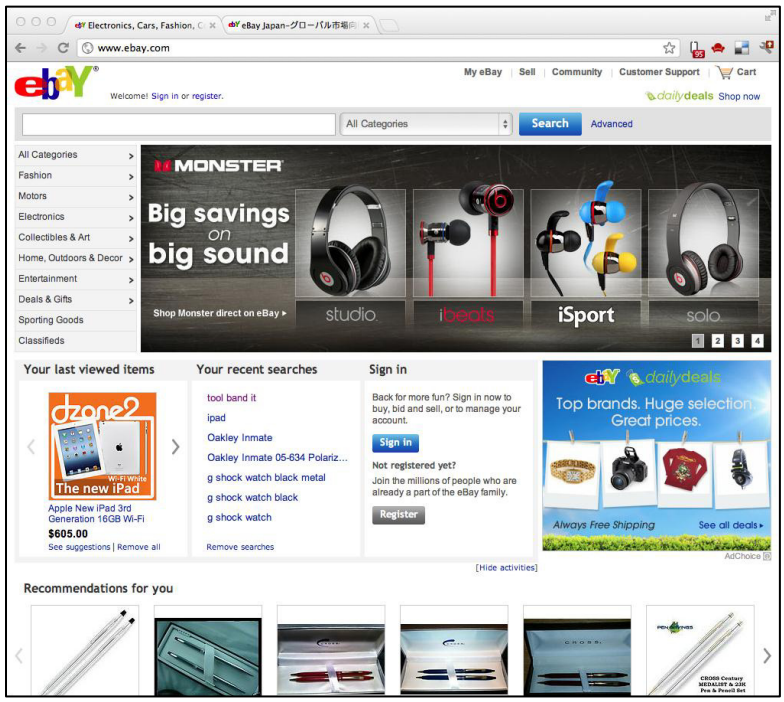
Na visão dos usuários, o design da interação é a tecnologia que faz os softwares serem fáceis de usar, eficientes e agradáveis. É o dispositivo que entende os objetivos e expectativas e ao mesmo tempo seu comportamento e lado psicológico (Wang, 2011).

Enquanto os usuários estão mais preocupados com a interface, possibilidades e facilidade de uso do sistema, os desenvolvedores levam em conta a escalabilidade do código, boa documentação e uso de frameworks, já os gerentes de empresas de softwares se importam apenas com a satisfação do cliente e o baixo valor de produção que resulta em mais lucros.

Além disso, ainda existem outras questões culturais que influenciam bastante na forma como os grupos de pessoas que, apesar de serem de um mesmo tipo, mas de diferentes regiões do estado ou país, consideram tal requisito mais importante como adequado ou não. Esse e outros problemas podem ser resolvidos utilizando IHC, visto que, atualmente, não é raro, uma empresa ou um desenvolvedor criar aplicativos para serem utilizados em todo o mundo.

Figura 2.7 - Website www.ebay.co.jp

Figura 2.8 - Website www.ebay.com

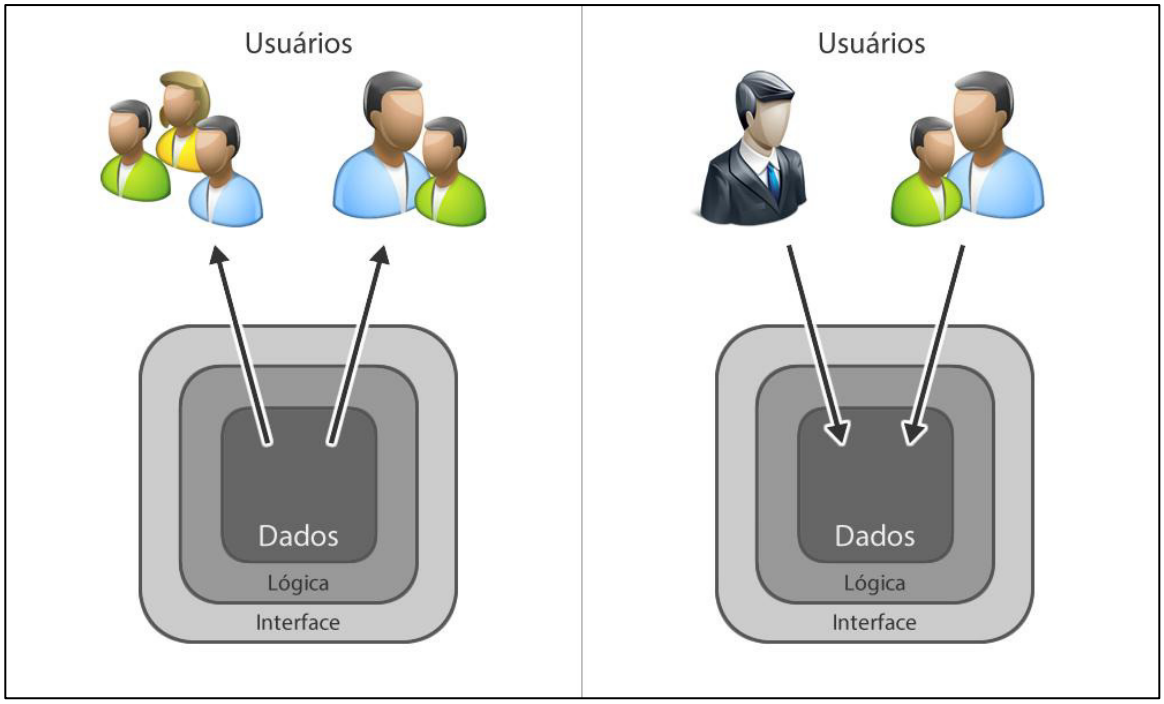


As Figuras 2.7 e 2.8 mostram um exemplo dessas diferenças culturais influenciando, nesse caso, dois websites. A Figura 2.7 mostra o site da empresa de vendas online Ebay em sua versão japonesa. Percebem-se diversas peculiaridades que podem parecer muito estranhas para pessoas ocidentais, tais como a ausência de menu na lateral superior esquerda, praticamente um padrão adotado na grande maioria dos sites de tal categoria que visam o público ocidental, e a barra de pesquisa no meio da tela, justificada à direita, diferentemente do padrão ocidental que tende a situá-la no topo justificada à esquerda. Já na Figura 2.8, podemos observar um exemplo típico de website de comércio eletrônico voltado para o público ocidental, o mesmo apresenta tanto a barra de busca quanto o menu nos locais padrões. Apesar de serem websites equivalentes, que possuem as mesmas funcionalidades e mesmo fim, possuem versões completamente diferentes adaptadas para seu público alvo de perfil específico.

É interessante notar que essas diferenças são de grande importância para a finalidade do site, que é possibilitar a compra e venda de mercadorias *online*. Como existe grande concorrência no setor, uma empresa que estuda e aplica técnicas de IHC nos seus sistemas tem um grande diferencial em relação aos demais e suas chances de fidelização do cliente são altíssimas.

Esse é um bom exemplo de uma nova abordagem que tem surgido com os avanços nos estudos de IHC, ilustrada pela Figura 2.9. Ao invés de iniciar o desenvolvimento focado nos dados e por fim levar em consideração a interface, esta tem ganhado cada vez mais importância e é um dos focos principais dos softwares modernos.

Figura 2.9 - Tipos de abordagem de desenvolvimento



Nessa nova abordagem, o projeto de um sistema interativo começa investigando os atores envolvidos, seus interesses, objetivos, atividades, responsabilidades, motivações, os artefatos utilizados, o domínio, o contexto de uso, dentre outros, para depois identificar oportunidades de intervenção na situação atual, a forma que a intervenção tomará na interface com o usuário e, finalmente, como o sistema viabiliza essa forma de intervenção (Barbosa & Silva, 2010).

Todavia, o sistema é apenas um, sempre interpreta ações de uma mesma forma, cabe aos desenvolvedores tratar as interações de forma que elas façam sentido para a grande maioria dos usuários, ou, em um mundo ideal, para todos. Uma das formas de tentar melhorar as possibilidades de interpretações das interações de um sistema é utilizar equipes multidisciplinares no desenvolvimento das mesmas (Barbosa & Silva, 2010). Dessa forma, cada um percebe e reflete sobre determinada interação de maneira diferente, o que

possibilita a facilidade em propôr um maior conjunto de ideias e compará-las em diversos aspectos, chegando a uma solução que englobe e satisfaça todos ou grande parte das percepções.

2.4 Engenharia Semiótica

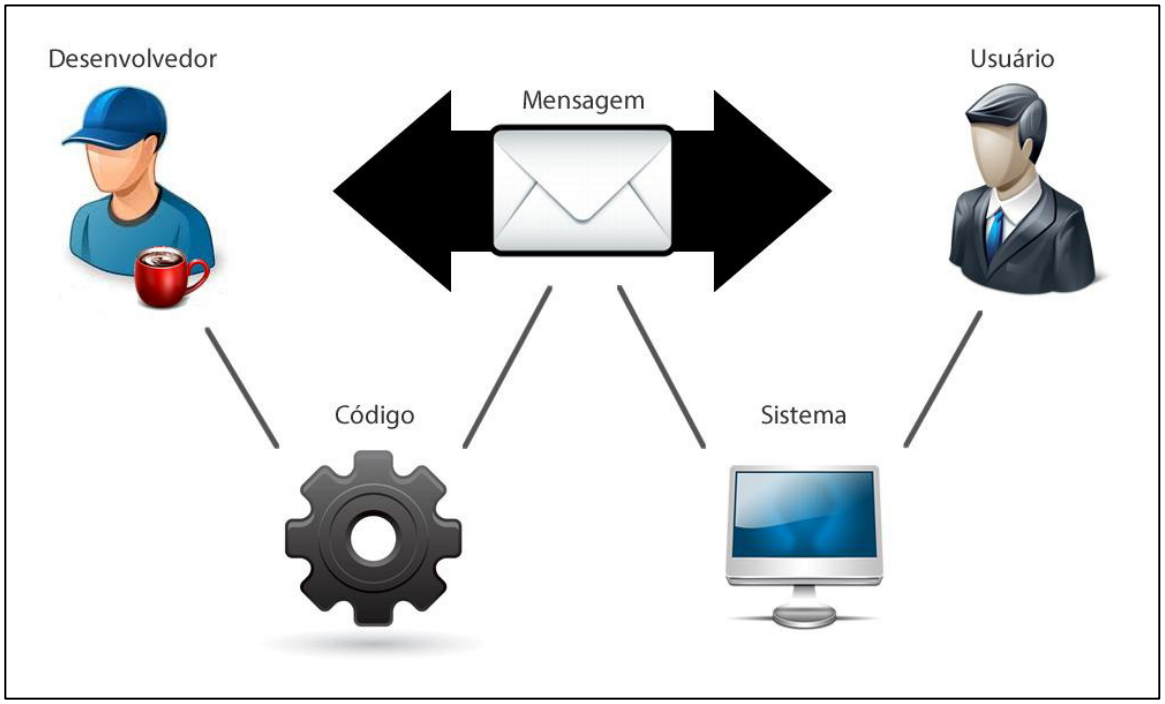
Outra forma de melhorar a usabilidade dos sistemas computacionais é utilizar a engenharia semiótica e suas diretrizes.

Segundo Souza (2005), a engenharia semiótica caracteriza a interação humano-computador como um caso particular de comunicação humana mediada por sistemas computacionais.

Na engenharia semiótica, o sistema é tratado como a comunicação do designer ou desenvolvedor para com o usuário. O intuito é fazer com que usuários e sistemas consigam se comunicar de forma eficiente e efetiva. Ou seja, que o usuário consiga compreender facilmente o que cada parte do sistema significa, qual o sentido de cada uma e como interagir com elas.

Um fator importante do processo de semiose é que este é fortemente influenciado pelo conhecimento prévio, pela cultura e pela experiência do usuário. Vem daí a importância de se estudar os usuários, suas atividades, experiências, valores e expectativas para conseguir fazer uma comunicação de forma mais eficiente das ideias que o sistema representa (Barbosa & Silva, 2010).

Figura 2.10 - Modelo do espaço de comunicação de Jakobson (1960) adotado na engenharia semiótica.



A Figura 2.10 mostra um resumo da perspectiva de engenharia semiótica em IHC, considerar o código como instrumento do desenvolvedor para se comunicar com o usuário através do sistema.

3 SOFTWARES DE AUTORIA

A forma convencional de desenvolvimento de aplicativos é baseada no uso de linguagens de programação tradicionais, a grande vantagem dessa abordagem é a liberdade dos autores para criar o software. Dessa forma, é possível, por exemplo, organizar o código conforme sua vontade, agrupar determinados trechos, especificar os tipos abstratos de dados e os tipos de interações a serem utilizados (Roddy, 1996). Como desvantagens, podem ser citadas a maior curva de aprendizado necessária para conseguir um bom resultado final, as preocupações com erros sintáticos, com as particularidades da linguagem utilizada e a dificuldade em criar abstrações que representem objetos do mundo real.

Os softwares de autoria surgiram para tentar diminuir as desvantagens do método tradicional de desenvolvimento.

Segundo Junior (2011), softwares de autoria são aqueles que visam desenvolver a criatividade do usuário que trabalha como autor de uma obra. A produção pode ser tanto a exposição de dados quanto a construção do conhecimento, dependendo das orientações do usuário. Com esse tipo de software os próprios usuários podem desenvolver suas aplicações sem que para isso precisem entender de programação de computadores.

Basicamente, um software de autoria é um programa que possibilita às pessoas que possuem pouco ou nenhum conhecimento em linguagens de programação e desenvolvimento de aplicativos poder criar aplicativos, animações, *ebooks* ou apresentações. Esse tipo de software normalmente encapsula uma linguagem de programação (é importante que mesmo que não seja utilizada pelos usuários, a linguagem de programação tradicional esteja presente no mesmo, pois, dessa forma, permite que os usuários avançados possam fazer ajustes finos na implementação) e um compilador (que transforma um programa escrito numa linguagem fonte para um programa escrito em uma

linguagem objeto) (Aho et al., 1995), permitindo que ao final do processo o usuário tenha um arquivo executável que realize as tarefas para as quais o software fora criado, de acordo com a vontade e a necessidade do usuário.

Esse tipo de aplicativo é muito importante, pois possibilita uma grande inclusão no mundo do desenvolvimento, permite que o processo de criação se torne mais simples, amigável e atrativo para leigos ou, até mesmo, iniciantes na área de desenvolvimento. Quanto maior a usabilidade e a capacidade de criar aplicativos com os mais diversos objetivos, desde os mais simples até aqueles mais complexos, melhor será considerado o software de autoria.

Atualmente, os softwares de autoria já têm um grande público alvo, são professores que utilizam como forma de suporte no processo de ensino, estudantes de ensino fundamental e médio que utilizam para realizar pequenas automações em robôs e, até mesmo, universitários que estão iniciando o aprendizado de programação.

Outras possibilidades de uso dos softwares de autoria são os próprios desenvolvedores quando precisam fazer rapidamente um protótipo de alguma aplicação, que, mesmo que tenha baixa fidelidade, possua a essência das interações da aplicação final.

Devido ao grande número de softwares dessa categoria, esse trabalho irá focar em apenas três. São eles: Scratch, Alice e Illumination Software Creator.

3.1 Scratch

Scratch é um ambiente de desenvolvimento que utiliza uma linguagem de programação baseada em blocos, projetada para ser intuitiva e fácil de ser utilizada por desenvolvedores iniciantes (Resnick, Kafai, Maeda, 2003). Scratch adiciona programação nas atividades de mídia rica e baseada em rede que são populares entre os jovens nos centros de ensino. Ele tira proveito do extraordinário poder de processamento dos computadores atuais, suporta novos paradigmas de programação e atividades que antes eram inviáveis, tornando ele uma melhor opção que as tentativas anteriores de introduzir a programação aos usuários iniciantes (Maloney et al., 2004).

Segundo Maloney et al. (2004), o objetivo desse trabalho é proporcionar aos seus usuários trabalhar em projetos significativos para as pessoas, tais como histórias animadas, jogos e arte interativa, que podem desenvolver fluência tecnológica, matemática, habilidades para resolver problemas e uma justificável autoconfiança que irá atendê-los bem nas mais diversas esferas de suas vidas.

Os desenvolvedores do Scratch tiveram a iniciativa de desenvolvê-lo devido à dificuldade de introduzir a programação para leigos ou jovens iniciantes, utilizando as linguagens de programação tradicionais. Essa foi a motivação para tentar expandir o acesso a esse tipo de atividade a grupos muito maiores de pessoas. Um fator que possibilitou essa ideia foi o fato de os computadores atuais terem grande poder de processamento, fato que inviabilizava o projeto anteriormente.

Seu desenvolvimento foi baseado em algumas diretrizes:

- O fato de os jovens reconhecerem o valor e potencial das ferramentas que têm a temática pelas quais eles se interessam;

- Jovens gostam de criar produtos finais que podem ser mostrados para outras pessoas;
- A ferramenta, uma ampla gama de diferentes tipos de temáticas, que tem apelo a jovens de diferentes culturas, idades, gênero e origem;
- Os jovens têm facilidade no uso de ferramentas novas;
- Os jovens podem aprender recursos adicionais de forma incremental;
- Os jovens têm a possibilidade de se aprofundarem na ferramenta para usar suas funções mais complexas (Maloney et al., 2004).

Scratch é baseado em uma estrutura de programação utilizando construção de blocos através do recurso de arrastar e soltar, que gera scripts ao colocar blocos gráficos unidos, como se o usuário estivesse montando um quebra cabeça (Maloney et al., 2008). Essa abordagem tende a eliminar erros de sintaxe, um dos maiores empecilhos nas linguagens de programação baseadas em texto, permitindo que o usuário foque no problema que ele deseja resolver. Existem diversas classe diferentes de blocos, cada classe reúne blocos com temática relacionada, como blocos de movimento, som, aparência etc. A Figura 3.1 mostra as diversas classes de funções disponíveis no Scratch, e algumas das funções da classe movimento.

Figura 3.1 - Classes de bloco do Scratch

Utilizando os diversos blocos das diversas classes, os usuários conseguem montar diversas listas, ou procedimentos que determinado objeto vai realizar. É possível executar pilhas em paralelo, e criar pilhas para diferentes objetos.

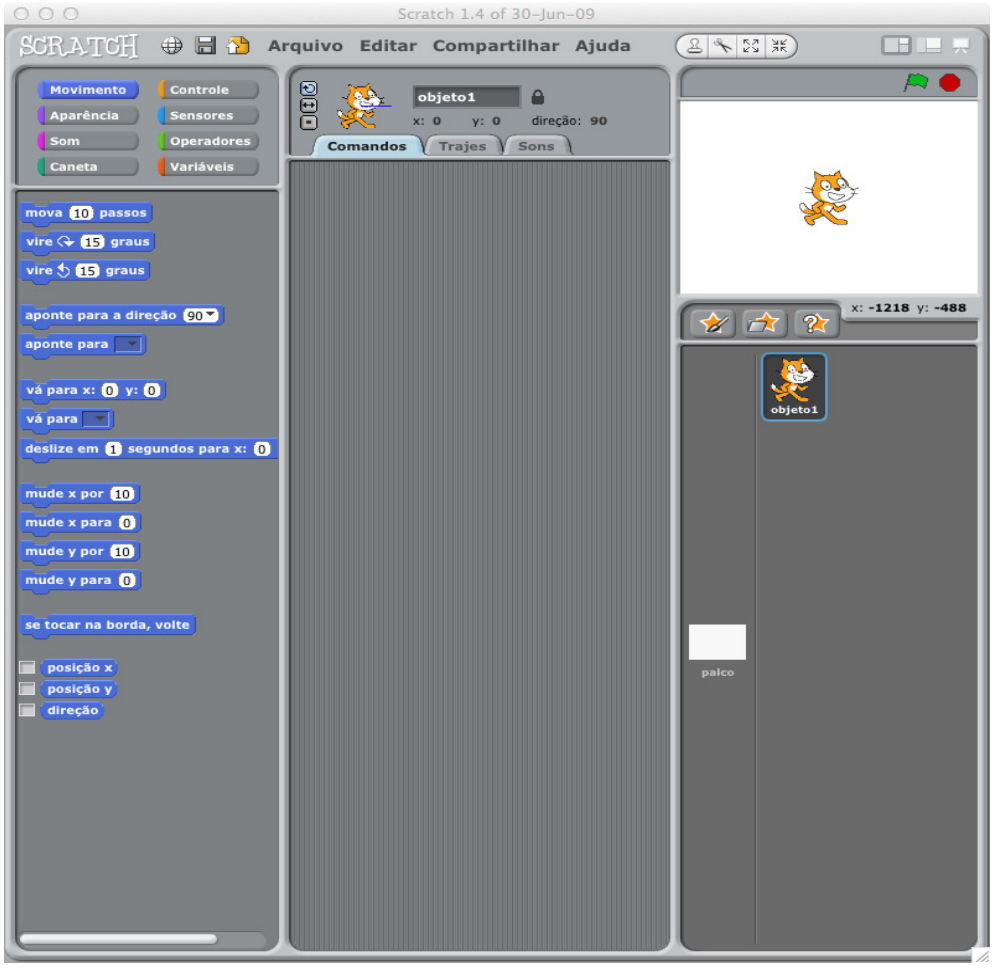
Utilizando essa abordagem os desenvolvedores conseguiram possibilitar que diversos elementos de mídia, como som e vídeo fossem incluídos na aplicação. Além de interações, operações matemáticas, iterações (laços) e etc. Tudo de maneira simples, e fácil de ser utilizado.

Outras funcionalidades interessantes do Scratch são a possibilidade de compartilhar facilmente o resultado da utilização da ferramenta na web e a de utilizá-lo como ferramenta de apoio à automação de uma placa de sensores,

chamada de Scratch Sensor Board e o Lego WeDo, plataforma de robôs programáveis da empresa Lego.

O Scratch é desenvolvido pelo grupo Kindergarten Lifelong, do laboratório Media MIT com os financiamentos de National Science Foundation, Microsoft, Intel Foundation, MacArthur Foundation, Google, Iomega, e Consórcio de Pesquisa MIT Media Lab (Scratch, 2011).

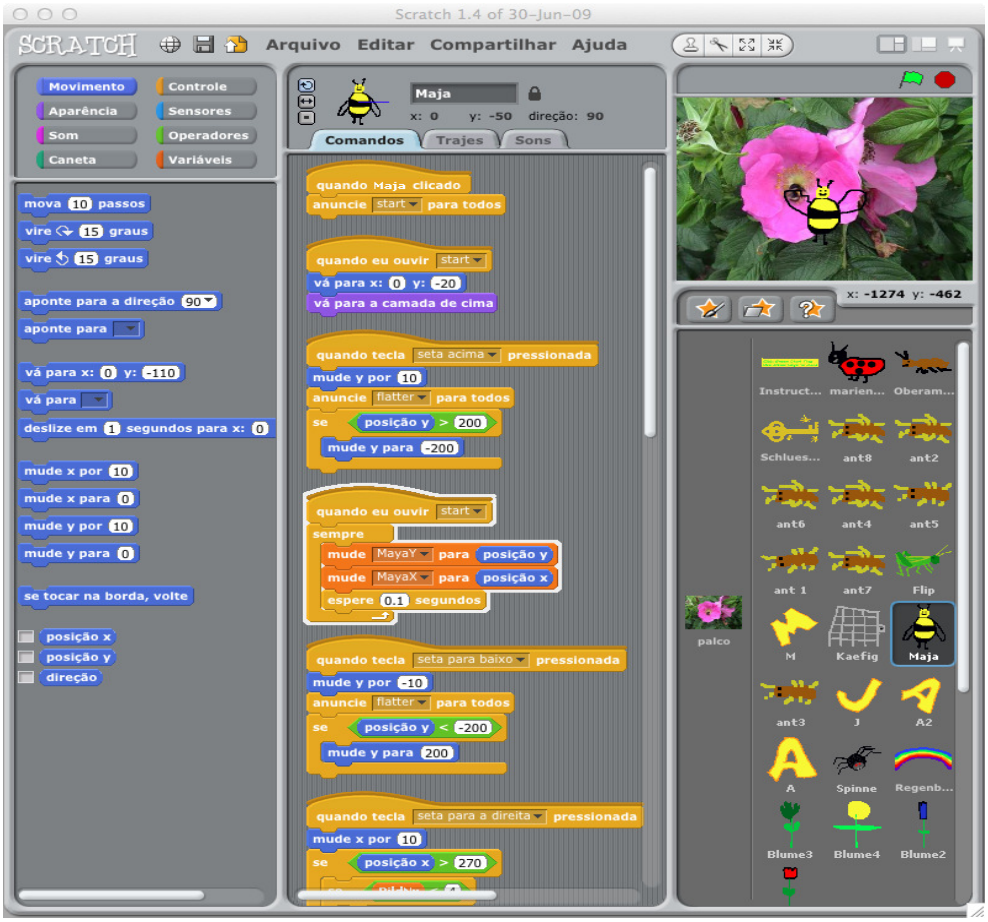
Figura 3.2 - Tela inicial do Scratch



Na Figura 3.2 podemos ver a tela inicial do Scratch, nota-se que seus menus e comandos são configurados automaticamente em português. No canto superior esquerdo podemos identificar as classes de blocos, na parte inferior

esquerda os blocos da classe movimento. Acima temos menus de controle da aplicação, e na parte central a área onde serão montadas as pilhas de comandos. No lado direito temos um *preview* das configurações gráficas e ao mesmo tempo a janela para simulações, na parte inferior direita temos um seletor de cenários e de objetos de trabalho.

Figura 3.3 - Exemplo Scratch



Na Figura 3.3 temos um exemplo de utilização do Scratch, nota-se que no console central do aplicativo existem várias pilhas de blocos com as mais diversas configurações. No canto superior direito é mostrado um *preview* da animação e logo abaixo fica localizado o menu de objetos e cenários.

3.2 Alice

Alice é um ambiente de programação 3D que torna fácil criar uma animação para contar uma história, jogar um jogo interativo ou compartilhar um vídeo na web. Ela permite que os usuários aprendam conceitos fundamentais da programação tradicional no contexto de criar vídeos animados ou jogos simples. Em Alice, os objetos 3D fazem parte de um mundo virtual no qual os usuários criam programas para animar esses objetos (Alice, 2012a).

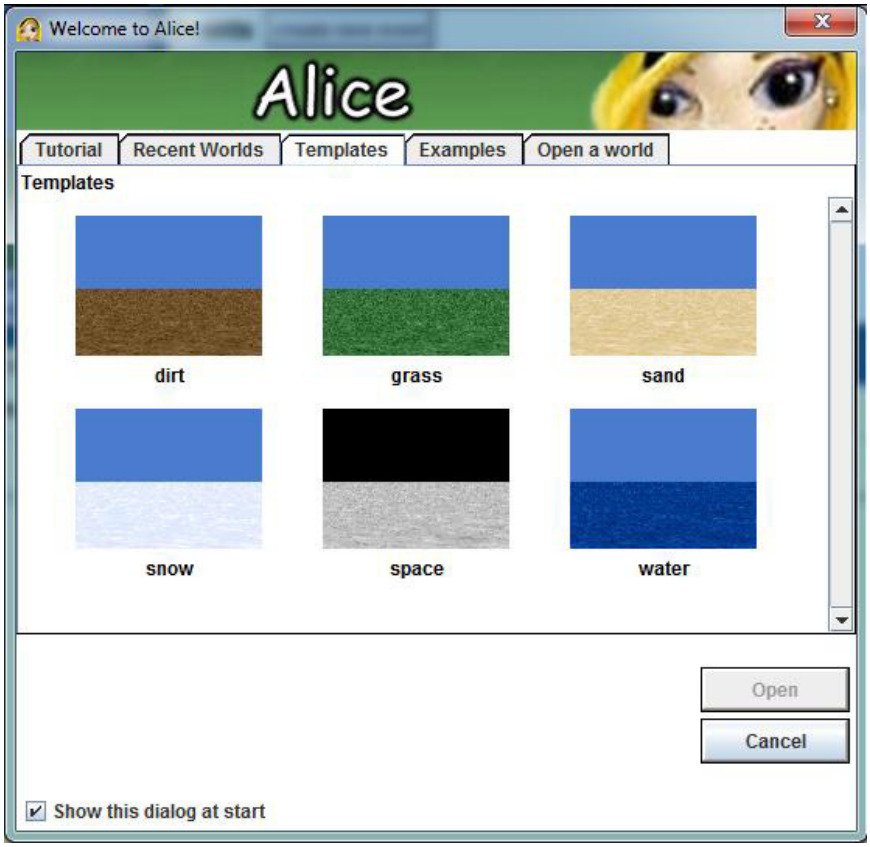
Segundo Conwa et al. (1999), o Projeto Alice foi iniciado com o objetivo de criar novas ferramentas que possibilitassem criar gráficos 3D de maneira mais acessível, de forma que um público maior pudesse ter acesso aos mesmos. Os princípios básicos foram:

- Escolher um público alvo e manter as suas necessidades em foco; no caso de Alice, estudantes de graduação de áreas não técnicas;
- Evitar notações matemáticas e enigmáticas na API (como, por exemplo, vetores e matrizes) sempre que possível e introduzir novas terminologias somente quando necessário;
- Fazer testes interativos do design com usuários reais, melhorando ambos - aprendizado e usabilidade do processo do sistema.

Assim como o Scratch, o Alice também é um software de autoria que se baseia em um formato de montar blocos e pilhas de códigos, mostrando sempre uma resposta visual do que determinado bloco faz, permitindo aos usuários relacionar a programação com os seus resultados. Os seus desenvolvedores afirmam que esse tipo de resposta rápida e eficiente tem uma grande porcentagem na atração de interesse pelo sistema. Seus testes mostram uma diminuição considerável na evasão de cursos de programação, além de melhorar o desempenho dos estudantes.

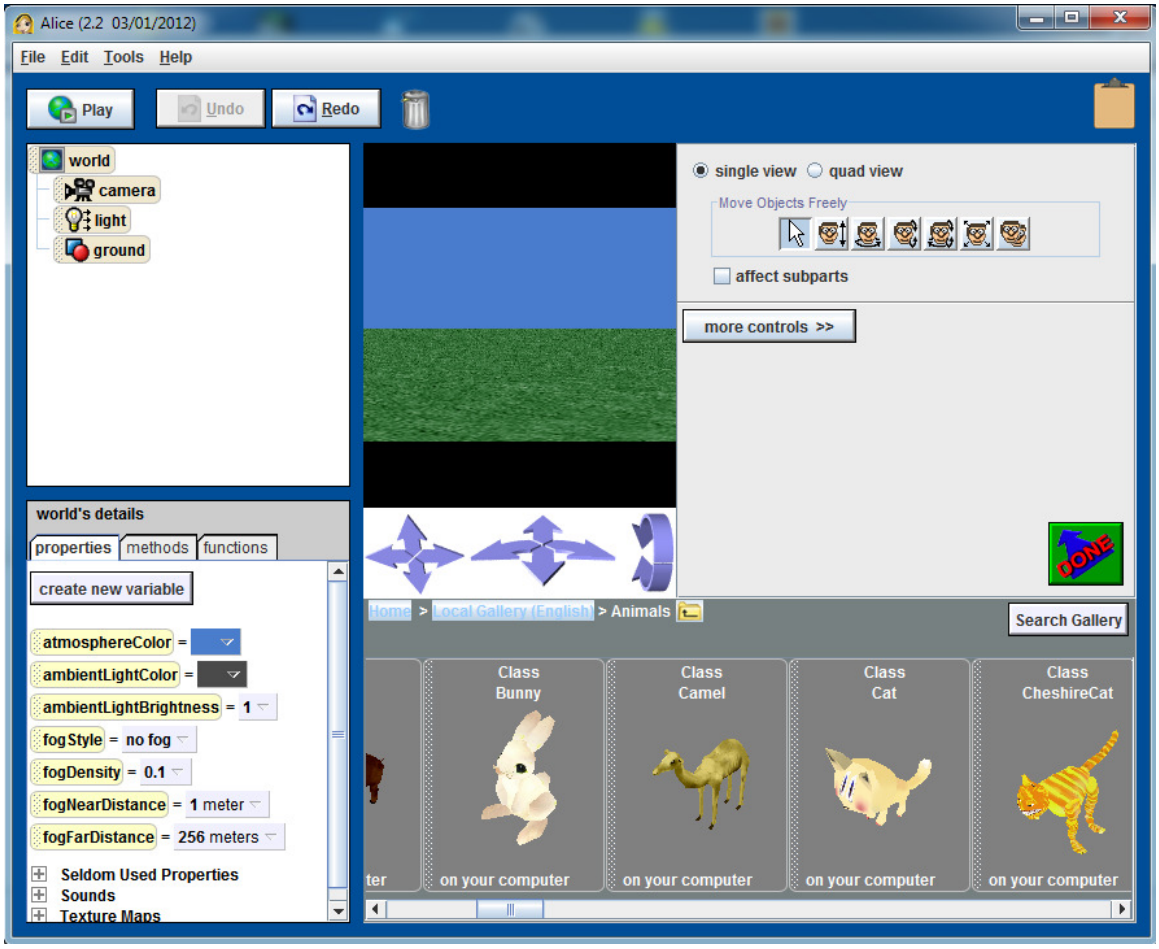
O Alice é desenvolvido pelos pesquisadores do Stage3, um grupo de pesquisa da Universidade Carnegie Mellone tem o financiamento da ElectronicArts, Oracle, DARPA, Intel, Microsoft, NSF, ONR, Google, Hyperion Books, Disney, The hearst Foundations e The Heins Endowments (Alice, 2012b).

Figura 3.4 - Tela Inicial Alice



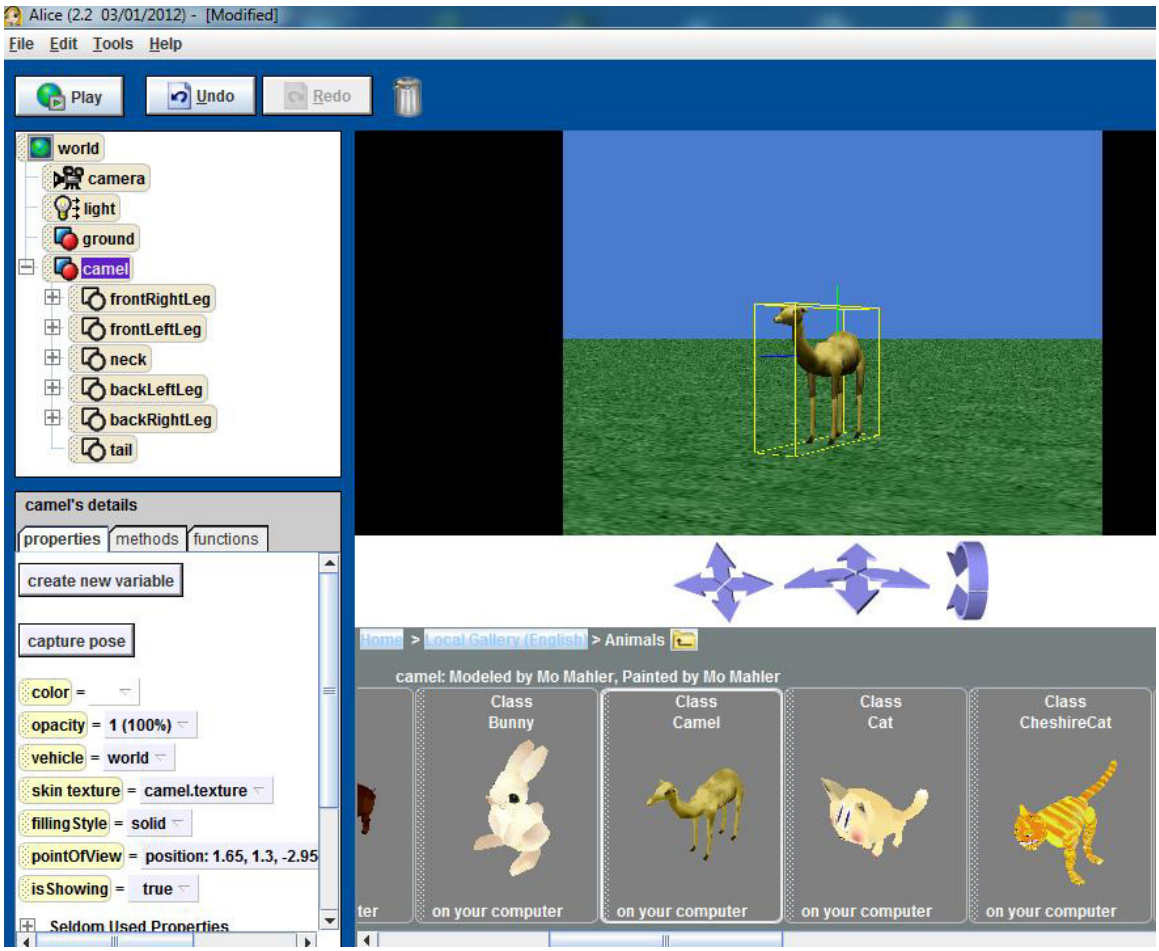
Como podemos ver na Figura 3.4, essa é a tela inicial do Alice, já é solicitado ao usuário que escolha um modelo (template), e várias outras configurações inicias do seu projeto.

Figura 3.5 - Tela de seleção de objetos do Alice



No lado esquerdo da Figura 3.5, temos na parte superior um resumo do seu World (ambiente de desenvolvimento) e na parte inferior, vários detalhes, possibilidades de funções, métodos e propriedades dos mais diversos objetos e cenários que podem fazer parte do seu mundo. No centro, existe uma tela que mostra um *preview* de suas ações ao mesmo tempo em que elas são efetuadas, por exemplo, ao selecionar um objeto para fazer parte do seu projeto

Figura 3.6 - Exemplo de objeto no Alice



Na Figura 3.6, pode-se perceber que, ao adicionar um objeto no seu projeto, automaticamente será gerada uma imagem do mesmo, e suas informações passam a ser acessíveis no menu superior esquerdo da aplicação.

3.3 Illumination Software Creator

O Illumination Software Creator (ISC), lançado em maio de 2010, segue a mesma linha dos softwares anteriores, é um programa de autoria, que permite ao usuário criar um produto final com pouco ou nenhum conhecimento em programação. No entanto, o ISC tem uma abordagem um pouco diferente, o usuário pode escolher qual será o destino da sua aplicação, na versão Free permite criar aplicações em HTML5 no formato de web apps, ou em Python para Desktop; já na versão Deluxe, ele suporta a criação de aplicativos em Adobe Flash para web, iOS e Android, além das possibilidades da versão Free (Lunduke, 2010).

O ISC permite que qualquer pessoa crie seus próprios softwares sem escrever uma única linha de código. A ideia é simples, organize blocos coloridos com as mais diversas funcionalidades da maneira que você quiser criando seu próprio software. Sem a necessidade de ler grandes livros de programação e sem curva de aprendizado (ISC, 2011).

O ISC também possui diversas classes de blocos agrupados de acordo com sua temática, nas Figuras 3.7 e 3.8 podemos ver os tipos de blocos disponíveis e uma das classes desses blocos respectivamente.

Figura 3.7 - Tipos de classes de blocos

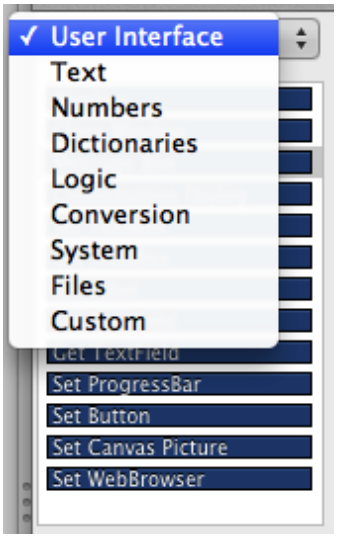
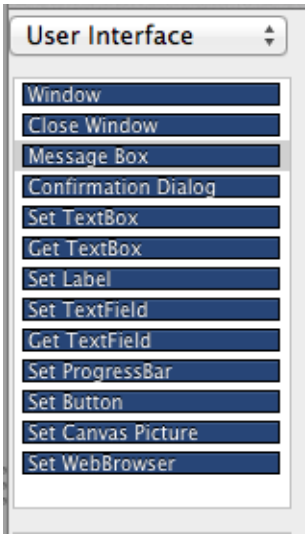
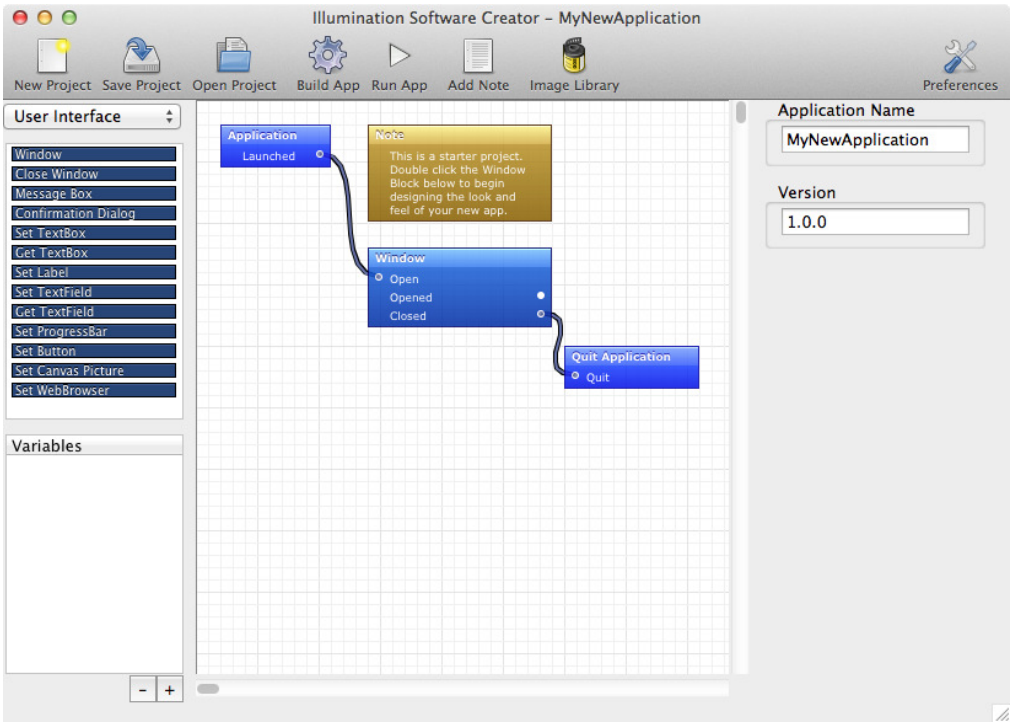


Figura 3.8 - Classe de blocos de interação com o usuário



A classe de blocos Custom serve para personalizar blocos de código permitindo o uso de programação tradicional no projeto. Ela está disponível apenas na versão Deluxe do ISC.

Figura 3.9 - Tela Inicial Ilumination Software Creator



Na Figura 3.9 podemos ver a tela inicial de interação com os usuários do ISC, acima existe um menu com estruturas de controle e preferências, na lateral esquerda da tela ficam as classes de blocos de código, seu seletor e o espaço onde ficam as variáveis criadas no decorrer do uso da aplicação. Na parte central fica a região de trabalho, onde os blocos serão agrupados conforme as necessidades do usuário. Na lateral direita fica localizado um menu auxiliar, variável de acordo com a estrutura selecionada no momento.

O ISC tem versões para Windows, Linux e Mac, além de um Live CD bootavel baseado no sistema open SUSE. O sistema tem uma versão comercial, cujo valor da licença é de US$49,99 com a possibilidade de desconto para a compra de várias unidades.

4 AVALIAÇÃO DE IHC

Conhecer, e seguir os critérios de qualidade e os diferentes processos de fabricação de um produto não garante que o resultado possua qualidade. Diversos problemas podem ocorrer, por exemplo alguma falha humana no processo, a baixa qualidade da matéria prima, etc. De certa forma essa linha de raciocínio também é válida para o desenvolvimento de softwares, onde os problemas costumam ocorrer nas etapa de coleta dos dados, interpretação ou até mesmo na implementação (Barbosa e Silva, 2010).

É difícil garantir a qualidade total de um produto seja ele qual for, seria necessário avaliar o produto final em todas as possíveis situações de uso, o que, além de inviável elevaria consideravelmente o tempo e custo do mesmo. Uma das formas de tentar conseguir um produto com ótima qualidade final é, claro, seguir os processos de design e desenvolvimento comprometidos com a qualidade mas também é preciso avaliar o produto final, se o mesmo atende ou não aos critérios de qualidade de projeto.

Existem diversas possibilidades de perspectivas na avaliação de um software: de quem o concebe, de quem o constrói e de quem o utiliza. Na perspectiva de quem constrói, os critérios de qualidade avaliados são por exemplo robustez e confiabilidade (Avizienis et al., 2004), nessa perspectiva o que ocorre fora do sistema (interações com os usuários) é abstraído e o foco são as relações que acontecem dentro do sistema. Existem vários métodos de avaliação para verificar a qualidade de um sistema utilizando esse ponto de vista, como por exemplo os testes de caixa-branca e caixa-preta (Pressman, 2002; Delamaro et al., 2007).

Essa avaliação costuma ser realizada através de uma série de testes durante todo o processo de desenvolvimento, tais como: testes de unidade, de sistema, de operação e de integração. Contudo, mesmo após realizar todos esses testes e confirmar que o software atende os seus requisitos e projeto, ele

ainda pode apresentar problemas de uso, pois as interações do mesmo ainda não foram testadas, ou seja, o ponto de vista do usuário ainda não foi levado em consideração. Problemas de IHC ainda podem continuar existindo depois que os problemas na construção já tenham sido solucionados (Barbosa e Silva, 2010).

Essa é a justificativa para a avaliação de IHC ser tão importante quanto os demais testes de qualidade. Nessa avaliação o foco passa a ser o que existe e acontece da interface com usuário para fora. E os critérios de qualidade avaliados são relacionados ao uso do software, como por exemplo: usabilidade, acessibilidade, comunicabilidade e etc.

As razões para realizar teste de IHC são as mais diversas. Segundo Prates e Barbosa (2003) identificar e corrigir os problemas relacionados com a qualidade de uso antes de o sistema ser entregue aos usuários demonstra profissionalismo da equipe de desenvolvimento. Já Tognazzini (2000) afirma que os problemas de IHC podem ser corrigidos antes e não depois de o produto ser lançado, e que essas correções permitem entregar um produto final mais robusto, que não precisará de correções logo.

O custo de avaliar a qualidade de uso não costuma ser alto em relação ao custo global de um projeto de desenvolvimento, principalmente quando os benefícios são considerados (Rubin, 1994; Bias e Mayhem, 2005).

A curto prazo os benefícios são o aumento de produtividade e satisfação dos usuários além da redução na quantidade de erros. A longo prazo, os custos com treinamentos e suporte diminuem.

Existem diversos métodos de avaliação de IHC, dentre eles pode-se citar a avaliação heurística, o percurso cognitivo, a inspeção semiótica e o teste de usabilidade que foi o método escolhido para ser utilizado neste trabalho. A seguir conheceremos um pouco de cada um desses métodos para então nos aprofundarmos nos estudos dos testes de usabilidade.

A avaliação heurística é um processo para encontrar problemas de usabilidade durante um processo de design criativo, esse método orienta os avaliadores a inspecionar sistematicamente a interface em busca de problemas que prejudiquem a usabilidade. Para isso são utilizadas algumas diretrizes de usabilidade chamadas por Nielsen de heurísticas (Nielsen e Molich, 1990; Nielsen, 1993). Algumas dessas heurísticas são: visibilidade do estado do sistema, controle e liberdade do usuário, consistência e padronização, prevenção de erros, reconhecimento em vez de memorização etc (Nielsen, 1994).

O percurso cognitivo tem como objetivo avaliar a facilidade de aprendizado de um sistema interativo, através da exploração de sua interface (Wharton et al., 1994). Esse método foi motivado pela preferência de muitas pessoas em aprender já com a utilização do sistema, em vez de participar de treinamentos, ler manuais ou utilizar a ajuda. Nesse método, o avaliador percorre a interface inspecionando as ações projetadas para um usuário, tentando se colocar no papel do mesmo e tenta detalhar como seria a sua interação com o sistema naquele momento (Barbosa e Silva, 2010). Em um sistema com projeto de IHC e boa usabilidade, espera-se que a própria interface guie os usuários pelas ações esperadas. Caso isso não ocorra, esse método possibilita a criação de hipóteses e possíveis soluções para os problemas.

O método de inspeção semiótica avalia a comunicabilidade de uma solução de IHC por meio da inspeção (Souza et al., 2006; Prates e Barbosa, 2007; Souza e Leitão, 2009). O objetivo desse método é avaliar a qualidade da emissão da metacomunicação do designer codificada na interface. Nele o avaliador realiza um conjunto de atividades de coleta de dados sobre experiências de uso e interpretação. Nelas o avaliador inspeciona a interface para identificar, interpretar e analisar os signos metalinguísticos, estáticos e dinâmicos nela codificados (Barbosa e Silva, 2010).

Por fim temos o teste de usabilidade que é um processo no qual participantes representativos avaliam o grau em que um produto se encontra em relação a critérios específicos de usabilidade (Rubin, 1994). Esse teste pode servir para os mais diversos propósitos que envolvem tipos de tarefas, medidas de performance e disposição de escalas, entrevistas ou inspeções a serem aplicadas buscando encontrar problemas de usabilidade e fazer recomendações no sentido de eliminar os problemas e melhorar a usabilidade do produto, ou com a finalidade de se comparar dois ou mais produtos (Ferreira, 2002).

Um teste de usabilidade permite responder perguntas do tipo: “Quantos erros os usuários cometeram nas primeiras sessões de uso?”, “Quantos usuários conseguiram completar com sucesso determinada tarefa?” e “Quantas vezes os usuários consultaram a ajuda on-line ou o manual de usuário?” (Barbosa e Silva, 2010).

Desses métodos citados anteriormente, somente o teste de usabilidade tem uma participação direta dos usuários, isto é, é um método de observação, diferente dos outros em que apenas os desenvolvedores ou designers, com o papel de avaliadores, aplicam o método, conhecido como métodos de inspeção. Como o objetivo deste trabalho é a comparação entre os três softwares de autoria, o teste de usabilidade foi escolhido por permitir tirar conclusões que os outros não permitem. Ele é o mais próximo da utilização final possível. Além disso, os demais métodos podem ser considerados testes de exploração ou de avaliação (são aqueles utilizados no início do processo de desenvolvimento, quando ainda não há um produto final pronto), ou seja, proporcionam uma avaliação formativa, enquanto o teste de usabilidade é, na maioria das vezes, um teste que pode ser considerado como de validação (realizado no final ou após o processo de desenvolvimento) do produto, e assim, proporciona a chamada avaliação somativa ou conclusiva (Barbosa e Silva, 2010).

O objetivo de uma avaliação de IHC é avaliar a solução do ponto de vista dos usuários, a melhor forma de fazer isso é justamente envolvê-los nesse

processo. Outra vantagem do teste de usabilidade é a possibilidade de fornecer mais dados quantitativos em relação aos demais métodos, fator importante para a comparação com os demais softwares de autoria. O Quadro 4.1 a seguir mostra essa informação.

Quadro 4.1 - Tipos de dados produzidos por método de avaliação

	Dados Quantitativos	Dados Qualitativos
Avaliação Heurística	+	+++
Percurso Cognitivo	+	+++
Inspeção Semiótica	+	+++
Teste de Usabilidade	+++	++

Fonte: (Barbosa & Silva, 2010)

4.1 Testes de Usabilidade nas ferramentas de autoria

Uma das formas de avaliar a qualidade da interação dos usuários com um software é usando o método de observação conhecido como teste de usabilidade. A aplicação desse teste possibilita ao avaliador obter informações e coletar dados sobre situações reais de uso do software. O registro e a análise desse tipo de teste possibilitam identificar grande parte dos problemas que o aplicativo possa apresentar. Diferente da avaliação por inspeção em que são identificados apenas os problemas potenciais que o software possa vir a apresentar.

O teste de usabilidade visa avaliar a usabilidade de um sistema interativo a partir das experiências de uso dos seus usuários-alvo (Rubin, 1994; Rubin e Chisnell, 2008).

É importante deixar claro que este método não estuda a interface em si, e sim a opinião dos usuários sobre a mesma (Nielsen, 1997). E a partir desse ponto são identificadas e tiradas conclusões sobre as melhores características de cada aplicação e a partir desses dados podemos construir uma proposta de

modelo descartando as características de menor qualidade e somando as de maior aceitação, que é o objetivo final da utilização dos testes neste trabalho.

Para realizar as medições desejadas, um grupo de usuários é convidado a realizar um conjunto de tarefas usando o sistema em um ambiente controlado. Durante a observação dessas experiências, foram registrados dados sobre o desempenho, a opinião e a satisfação de uso dos usuários.

Em um teste de usabilidade 4 a 5 participantes serão capazes de expor 80% das deficiências de usabilidade de um produto, ou seja, se o objetivo dos testes forem tentar expor os principais problemas de usabilidade do produto, essa quantidade de participantes possibilita que os resultados sejam alcançados rapidamente e que uma vasta maioria dos problemas de usabilidade sejam encontrados (Ferreira, 2002).

É importante que o teste seja conduzido dentro de um rigor experimental, que pode ser alcançado mantendo a consistência das sessões, registrar se algum problema incomum ocorrer, realizar um teste piloto, manter a simplicidade do teste e tornar o mesmo o mais próximo possível da realidade (Ferreira, 2002).

Normalmente os testes de usabilidade tem uma declaração dos possíveis problemas a serem validados com o teste, contudo como a abordagem nesse trabalho foi de certa forma diferenciada, os testes não possuíram essa declaração. Isto serviu inclusive para torna-los mais imparciais.

4.2 Personas

Para conseguir resultados mais específicos: ao agrupar os tipos de usuários semelhantes, é mais fácil analisar e tirar conclusões dos testes; e imparciais: ao utilizar representantes de usuários reais as interferências no processo diminuem (Cooper, 1999). Nesse estudo, foram feitos testes com 3 grupos de usuários. Cada grupo será representado por uma Persona.

Uma Persona é um personagem fictício, arquétipo hipotético de um grupo de usuários reais, criada para descrever um usuário típico (Pruitt e Adlin, 2006).

Segundo Cooper et al.(2007), uma persona é uma descrição precisa de características de um grupo de usuários aos quais o sistema vai atender. O conceito de Persona normalmente é utilizado para apoiar o desenvolvimento de aplicações, pois ajuda a manter o foco nas necessidades principais dos usuários. Contudo, este conceito foi utilizado neste trabalho por ser uma boa forma de representação de cada grupo de usuários.

Cada Persona deve ter identidade, ou seja, nome, idade, e outras informações básicas; objetivos: quais os objetivos dessa pessoa ao utilizar um computador ou software específico; habilidades: as especialidades e competências; tarefas: quais as tarefas básicas que essa persona realiza ao utilizar o computador ou software (Courage e Baxter, 2005).

4.3 Tipos de personas utilizados

4.3.1 Persona do tipo 1

José Silva, estudante de Ciência da Computação - “eu gosto mesmo é de programar”.

José Silva, 20 anos, estudante do 7º período de Ciência da Computação, já trabalhou como técnico de suporte e agora é estagiário no setor de desenvolvimento de softwares de uma empresa multinacional.

Tem plenos conhecimentos em programação, utiliza e domina diversas linguagens, além disso, tem conhecimentos aprofundados nas áreas de redes e banco de dados.

Utiliza o computador tanto no trabalho quanto em casa, já foi usuário do Windows, mas agora utiliza Linux. Passa em média 7h/dia utilizando seu notebook, tanto para trabalho quanto para diversão.

4.3.2 Persona do tipo 2

Maria Silva, estudante de Desenho Industrial – “o mais importante são as cores”.

Maria Silva, 23 anos, estudante do 8º período de Desenho Industrial, já participou de um projeto de criação de websites para diversos eventos que acontecem na sua cidade. Atualmente trabalha em uma empresa de publicidade, é responsável por coordenar a equipe de design.

Tem conhecimento razoável na área de computação, no entanto não sabe programar.

Utiliza o computador no trabalho para checar e-mails, fazer pesquisas, agendar reuniões, planejar ações, etc. Já em casa e na universidade, seu uso é para lazer e para realizar as atividades acadêmicas. Utiliza, em média, o computador por 3h/dia.

4.3.3 Persona do tipo 3

Pedro Sousa, Auxiliar Administrativo – “se tem mais de 3 botões já é complicado”.

Pedro Sousa, 30 anos, formado em administração, é funcionário público de uma secretaria da sua cidade. Exerce a função de auxiliar administrativo há 10 anos.

Tem pouco ou nenhum conhecimento na área de computação, utiliza apenas funções básicas do seu computador, mesmo assim, tenta evitar o máximo possível.

Pedro utiliza o computador, em média, 6h/semana apenas para questões relacionadas ao trabalho, normalmente responder e-mails ou apenas para imprimir algum documento.

4.4 Plano de ação do teste de usabilidade

As atividades que constituíram os testes de usabilidade estão descritas no Quadro 4.2 a seguir.

Quadro 4.2 - Atividades dos testes de usabilidade

Atividade	Tarefa
Preparação	• **Definir o perfil dos participantes** • **Preparar material para observar e registrar os testes** • **Executar um teste piloto**
Coleta de dados	• **Observar e registrar as informações relevantes durante os testes**
Interpretação / Consolidação	• **Reunir, contabilizar, sumarizar, processar os dados coletados**
Relato dos Resultados	• **Relatar os resultados da interpretação dos testes**

Na atividade de preparação são realizadas as tarefas comuns aos métodos de avaliação por observação, dentre elas a definição dos tipos de perfil dos participantes (item 4.2), a preparação dos materiais necessários para a melhor fluidez do teste, como formulários, tabelas, etc. E para finalizar, é realizado um teste piloto, que serve como forma de garantir que todos os requisitos para a realização do teste de forma eficiente estejam cumpridos.

Os testes foram realizados por 6 pessoas de cada tipo de persona, cada pessoa do grupo em uma determinada ordem diferente de utilização dos 3 aplicativos. Essa foi a forma utilizada para garantir que mesmo que a utilização de um aplicativo pudesse influenciar na experiência de uso do seguinte,

haveriam 2 pessoas que testaram cada aplicativo sem influência de testes anteriores, além disso, dessa forma foi possível realizar uma quantidade razoável de testes sem a necessidade de muitas pessoas diferentes envolvidas. O Quadro 4.3 ilustra a ordem em que cada persona foi submetida ao teste de cada software.

Quadro 4.3 - Ordem de utilização de cada grupo de 6 Personas

Usuários	1º Aplicativo	2º Aplicativo	3º Aplicativo
1º	Scratch	Alice	ISC
2º	Scratch	ISC	Alice
3º	Alice	Scratch	ISC
4º	Alice	ISC	Scratch
5º	ISC	Scratch	Alice
6º	ISC	Alice	Scratch

A coleta de dados consiste no questionário pré-teste, a sessão de observação e a entrevista pós-teste. Essa coleta pode ser considerada o teste de usabilidade em si, e acontece da seguinte forma: primeiramente é feita uma apresentação do teste com a utilização do roteiro apresentado na Figura 4.1.

Figura 4.1 - Roteiro de apresentação do teste

Teste de usabilidade – Softwares de autoria

Olá, meu nome é Leandro Lima, sou estudante de Ciência da Computação da UFMA e iremos trabalhar juntos nesta sessão de teste.

Estaremos efetuando o teste de um software de autoria disponível no mercado, seu nome é: Scratch, Alice, ISC.

Todas as informações coletadas com esses testes permanecerão anônimas.

O teste terá duração de aproximadamente 20 minutos, é importante que você utilize o aplicativo natural e tranquilamente, como se estivesse utilizando qualquer outro. Além disso, é importante que você diga o que está pensando durante a execução de qualquer tarefa, pois todo tipo de informação é útil nesse trabalho. O teste poderá ser encerrado precocemente se você sentir necessidade, basta informar.

Você poderá fazer perguntas, mas eu não poderei respondê-las. Isto irá ocorrer porque preciso verificar como você irá trabalhar com o software de forma independente.

Faça o melhor e não se preocupe com os resultados. São os aplicativos que estão sendo testados e não você. Sua contribuição poderá ajudar a melhorá-los.

Eu me sentarei próximo a você para tomar algumas notas e registrar observações.

Você irá também responder a alguns questionários. É importante que sejam utilizadas informações verdadeiras e sinceras no preenchimento dos mesmos.

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

Você pode decidir invalidar seus dados, desde que me comunique até o final do teste. Neste caso, seus dados e resultados não constarão do processo de análise do teste.

Sua integridade será totalmente preservada, pois seus dados e identificação serão utilizados apenas para posterior análise dos testes por pessoal autorizado.

Você tem alguma pergunta?

Se não, você pode começar a utilizar o sistema livremente que o teste será iniciado.

Agradecemos por sua colaboração.

Em seguida, os usuários respondem um questionário a fim de serem identificados e agrupados conforme suas características. Este formulário pode ser observado nas imagens 4.2, 4.3, 4.4, 4.5 e 4.6.

Figura 4.2 - Introdução do questionário

Questionário para identificação do perfil dos participantes do teste de usabilidade

O objetivo deste questionário é colher informações sobre o perfil do participante do teste de usabilidade a ser realizado utilizando um software de autoria.

Ao concordar em preencher este formulário esteja ciente que as informações fornecidas são vitais para a conclusão dos testes e as mesmas podem ser utilizadas e reproduzidas de acordo com a necessidade do avaliador.

Nas questões de marcar, favor assinalar somente 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.

A Figura 4.2 faz uma pequena introdução sobre o questionário, que tem prosseguimento com as figuras a seguir:

Figura 4.3 - Perguntas sobre informações pessoais

1) Informações Pessoais

1. Qual seu nome? _____
2. Qual a sua idade? _____ anos.
3. Sexo: M [ ] F[ ]

Figura 4.4 - Perguntas sobre informações educacionais

2) Informações Educacionais

1. Qual seu grau de instrução?
    a. Segundo grau incompleto
    b. Segundo grau completo
    c. Terceiro grau incompleto
    d. Terceiro grau completo

Escreve o nome do curso que está fazendo ou que completou de acordo com o grau assinalado acima: _____

Figura 4.5 - Perguntas sobre informações profissionais

3) Experiência Profissional

1. Qual a sua profissão? _____
2. Há quanto tempo se encontra nesta profissão?
    a. Menos de 1 ano
    b. Entre 1 ano e 2 anos
    c. Entre 2 anos e 4 anos
    d. Mais de 4 anos

Figura 4.6 - Perguntas sobre a experiência computacional

4) Experiência Computacional

1. Há quanto tempo você utiliza computador?
    a. Menos de 1 ano
    b. Entre 1 ano e 2 anos
    c. Entre 2 anos e 4 anos
    d. Mais de 4 anos
2. Em que local você utiliza o computador? (Pode-se marcar mais de uma opção)
    a. Em casa
    b. No trabalho
    c. Na universidade
    d. Outro: _____
3. Em média, quantas horas por semana você utiliza o computador?
    a. Menos de 2 horas
    b. Entre 2 e 5 horas
    c. Entre 5 e 10 horas
    d. Mais de 10 horas
4. Ao utilizar o computador, quais atividades você mais realiza?
    a. Email
    b. Navegar na internet
    c. Redes sociais
    d. Programação
    e. Outro:_____
5. Ao utilizar o computador, quais aplicativos você mais utiliza?
    a. Browser
    b. Word
    c. IDEs
    d. Outro:_____
6. Você sabe o que é um software de autoria?
    a. Sim [ ]
    b. Não [ ]

    Se sim, qual(is) você já utilizou? _____

Em seguida, o mesmo utiliza o sistema por um período de tempo, normalmente entre 15 e 20 minutos. Durante essa utilização, o avaliador faz a coleta de dados principais, utilizando o formulário representado na Figura 4.7.

Figura 4.7 - Formulário de coleta de dados

Coleta de Dados Pelo Avaliador

Dia:_____ Hora:_____ Nome do Avaliado:_____

Software Utilizado:_____

Tempo até a conclusão (minutos):	
Quantidade de utilizações de ajuda:	
Quantidade de erros do avaliado:	
O usuário conseguiu algum resultado final?	
Quanto tempo demorou para a produção do 1º resultado final?	
Tempo gasto na ajuda (minutos):	
Quantidade de utilizações de ferramenta de desfazer/voltar:	
O usuario utilizou as principais funções do aplicativo?	

O usuário encontrou dificuldade para entender alguma interação do aplicativo mesmo com a utilização da ajuda? Qual?

_____

Quais as maiores dificuldades o usuário apresentou na utilização deste software?

_____

Foram observados falsos affordances? Quais?

_____

Comentários dos usuários:

Comentários e observações do Avaliador:

E, por fim, é feito um questionário final para saber a opinião do usuários sobre o sistema testado, como pode ser observado na Figura 4.8.

Figura 4.8 - Entrevista final

Entrevista Final

Nota do usuário para o software: (0 a 10)	
Qual a sua opinião sobre o software testado? (1 palavra)	
Qual a nota para a usabilidade, facilidade de uso? (0 a 10)	

Quais as maiores dificuldades encontradas na utilização deste software?

_____

Consegue imaginar a utilização desse software no seu dia a dia? (S/N)

_____

Você notou algum erro no software? Qual(is)?

_____

Na sua opinião, quais os pontos fortes e fracos deste software?

_____

Você ficou satisfeito com a utilização do software ou frustrado?

_____

Esse aplicativo se assemelha com algum outro que você já tenha usado? Qual?

_____

O que você mudaria no software testado?

_____

Na atividade de interpretação ou consolidação, os dados são organizados de forma a evidenciar a relação entre os mesmos, visando chegar às conclusões que objetivaram os testes. Esses dados serão apresentados na forma de gráficos, tabelas, médias e quaisquer outros indicadores relevantes para esse estudo. Nessa fase, segundo Kauniavsky (2003), os testes de usabilidade também fornecem dados qualitativos, em que é possível identificar a origem, o motivo dos problemas de usabilidade, de interação de cada usuário. Cabe ao observador e analisador verificar todos os dados coletados para conseguir chegar a explicações para os problemas que tenham surgido.

Nessa etapa, os dados também devem ser categorizados, de forma a facilitar ao aplicador dos testes que exponha suas conclusões e teorias sobre o motivo de cada problema ou facilidade no sistema (Kuniavsky, 2003).

O relato dos resultados deve descrever: tabelas e gráficos que sumarizam as medições realizadas e uma lista dos problemas encontrados indicando local, descrição, justificativa e sugestão de solução (Barbosa e Silva, 2010). Além disso, como o objetivo desse trabalho é comparar três sistemas, nos relatos, sempre que possível, serão feitas comparações entre os mesmos.

Além disso, todos os testes foram realizados utilizando um notebook com a seguinte configuração: Processador Intel Core i5 2.3GHZ, 6gb de memória RAM, tela de 15” com resolução de 1440x900, teclado em português com ç.

Os softwares testados foram nas suas respectivas versões estáveis mais recentes.

4.5 Objetivos dos testes

O objetivo dos testes foi avaliar as seguintes metas de usabilidade:

- Usos da ajuda do aplicativo e a qualidade da mesma;
- Tempo necessário para obter o primeiro resultado final;
- Chegar ao resultado com menos tentativas fracassadas, sem ajustes;
- Quantidade de funções do aplicativo utilizadas;
- Quantidade de erros encontrados;
- Uso da ferramenta desfazer, voltar;
- Tempo de realização de determinada tarefa.

Esses objetivos foram detalhados em perguntas de forma que ficassem operacionais conforme indica Sharp et al. (2007).

4.6 Relato dos resultados dos Testes de Usabilidade

Durante os testes, ocorreram 12 desistências, ela aconteceram por volta de 5 minutos do início dos mesmos e se deram conforme o Quadro 4.4.

Quadro 4.4 - Desistências conforme tipo de Persona

	Persona 1	Persona 2	Persona 3	Total
Scratch	0	0	0	0
Alice	0	2	3	5
ISC	0	3	4	7
Total	0	5	7	**12**

Tais resultados se devem ao fato da maior dificuldade das Personas do tipo 3 em utilizar aplicativos diferentes dos quais estavam habituados e à maior complexidade de utilização do ISC.

Quadro 4.5 - Quantidade média de erros por avaliado

	Persona 1	Persona 2	Persona 3	Total
Scratch	0,33	0,50	0,50	0,44
Alice	0,17	0,00	0,33	0,17
ISC	2,33	0,67	0,33	1,11
Total	0,94	0,39	0,39	**1,72**

Foram considerados erros, sempre que um usuário tentou realizar uma função não permitida pelo software ou quando o mesmo conseguiu finalizar a interação mas a mesma não produzia o efeito desejado, a quantidade de erros pode ser observada no Quadro 4.5, que mostra a média de erros por avaliado para cada grupo de personas e para cada software. Os principais erros foram: no Alice a tentativa de voltar para a tela do aplicativo sem antes fechar a janela de saída, Figura 4.8; já no ISC houveram mais erros, a tentativa de rodar a aplicação sem adicionar campos obrigatórios como pode ser visto na Figura 4.9

ou com blocos sem as ligações necessárias e a utilização de determinada variável com o tipo errado e no Scratch a tentativa de conectar bloco incompatíveis, representado pela Figura 4.10, e a adição de blocos em objetos errados.

Figura 4.9 - Erro Alice

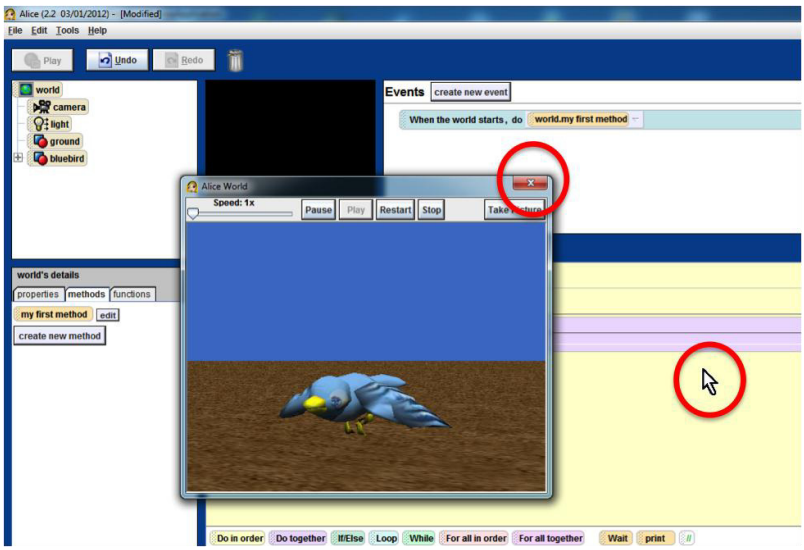


Figura 4.10 - Erro ISC

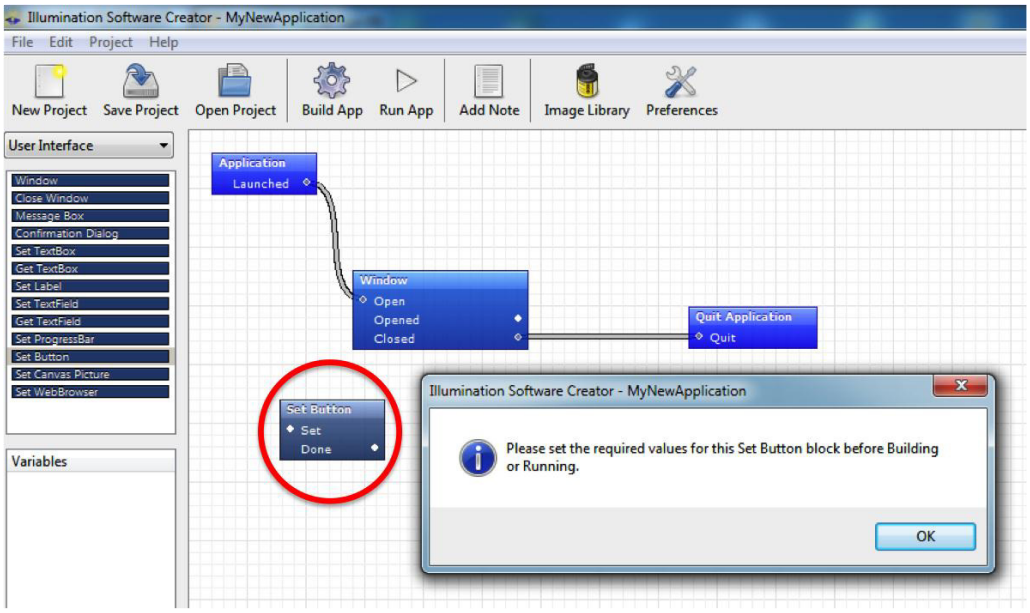


Figura 4.11 - Erro Scratch

Os motivos de as Personas 1 terem mais erros é reflexo da utilização e exploração de mais e mais complexas funcionalidades em comparação com as demais.

Em relação ao uso de ajuda dos aplicativos, foram coletadas duas informações, o tempo aproximado médio de utilização de ajuda Quadro 4.6 e a quantidade de avaliados que utilizou esse recurso Quadro 4.7.

Quadro 4.6 - Tempo médio em minutos de utilização de ajuda

	Persona 1	Persona 2	Persona 3	Total
Scratch	0,00	0,50	0,17	0,23
Alice	0,00	0,33	0,33	0,22
ISC	0,00	0,17	0,17	0,11
Total	0,00	0,33	0,22	**0,18**

Quadro 4.7 - Quantidade de avaliados que utilizou os menus de ajuda

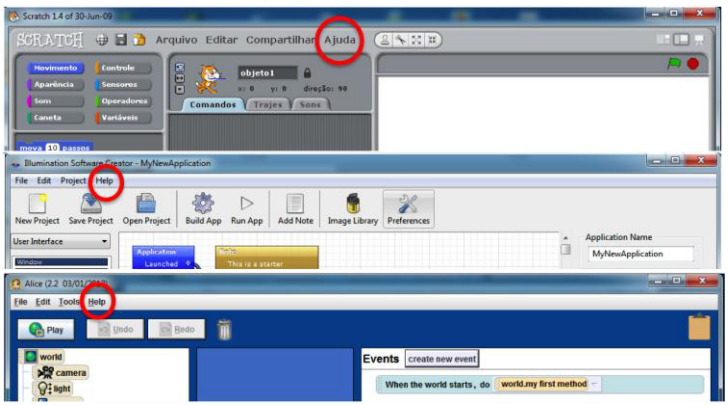
	Persona 1	Persona 2	Persona 3	Total
Scratch	0	4	1	5
Alice	0	3	2	5
ISC	0	2	1	3
Total	0	9	4	**13**

Essas informações mostram que as personas 1 não tem o costume de utilizar a ajuda de aplicativos, não importando qual seja. Já as personas do tipo 2 foram as que mais utilizaram e por mais tempo.

Porem os menus de ajuda dos aplicativos não foram muito efetivos, provavelmente este foi o motivo de terem sido pouco utilizados, o Alice por exemplo, sua ajuda é constituída de diversos tutoriais práticos, assim como o ISC. O Scratch é o único que mostra as ferramentas e suas funcionalidades uma a uma. Além disso, no Scratch o menu de ajuda fica mais centralizado, com mais destaque em comparação com os demais, detalhes vistos na Figura 4.11.

Uma critica em relação à ajuda do Scratch é que apesar de o software ser o único com menus em português, sua ajuda esta disponível somente em inglês, bem como a dos demais.

Figura 4.12 - Comparação entre acesso aos menus de ajuda



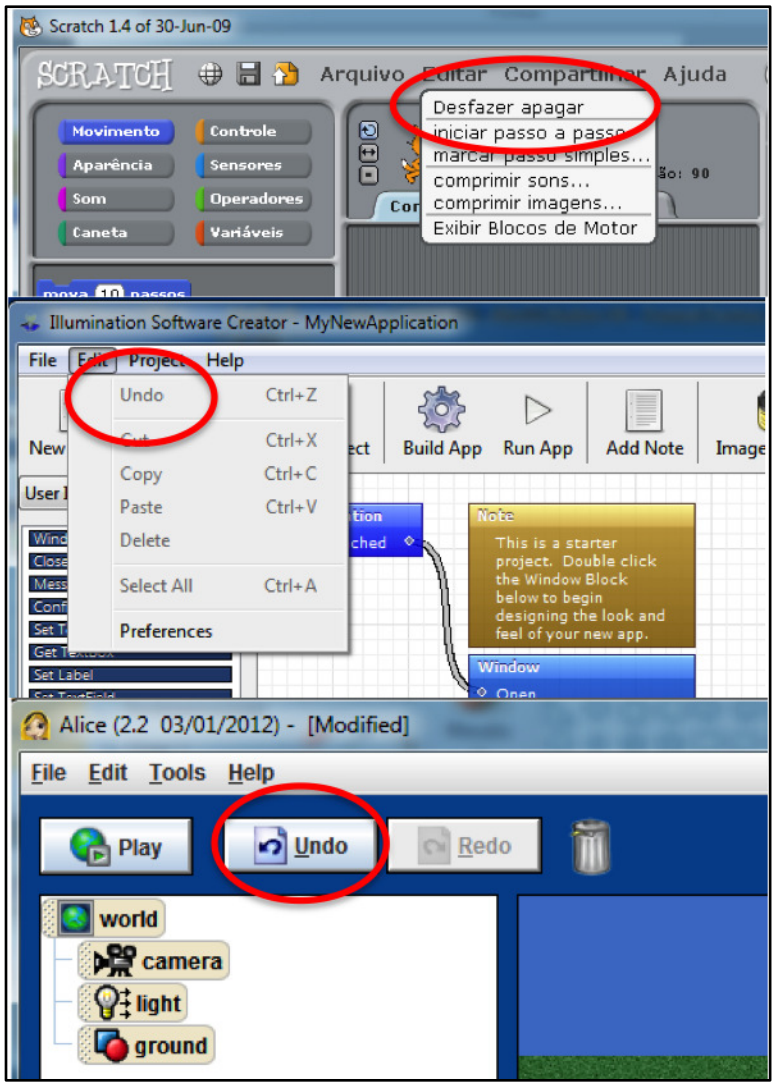
A utilização do comando desfazer, que pode ser observada no Quadro 4.8, foi maior com as personas 1, isso se deu ao fato de as mesmas terem explorado mais os aplicativos, utilizado funcionalidade mais complexas. É interessante observar que o ISC não possui essa funcionalidade, nem mesmo os comandos “Ctrl+z” funcionam, apesar de ser visível na sua interface. Já a quantidade maior de utilizações no Alice em relação ao demais é justificável por diversos fatores: o Alice possui um botão na interface principal com essa funcionalidade e os comandos “Ctrl+z” funcionam nele; já no Scratch, quando os usuários sentiam a necessidade de desfazer o software apenas possibilita desfazer o último comando de apagar, mas mesmo se fosse diferente os usuários ainda usariam pouco, pois no Scratch é bem simples apagar o que você fez e fazer novamente.

Quadro 4.8 - Quantidade de utilizações da função "desfazer"

	Persona 1	Persona 2	Persona 3	Total
Scratch	2	1	0	3
Alice	14	7	4	25
ISC	0	0	0	0
Total	16	8	4	**28**

A Figura 4.12 mostra como são apresentador aos usuários as opções de desfazer de cada software.

Figura 4.13 - Apresentação da funcionalidade "desfazer" em cada software



A métrica seguinte, ilustrada no Quadro 4.9, mostra o tempo que os usuários necessitaram para criar o primeiro resultado, foi considerado resultado qualquer saída do aplicativo com algum objeto ou interação. Com esses dados fica claro que as personas do tipo 1 tiveram mais facilidades para criar resultados nos 3 aplicativos testados. E ao mesmo tempo mostra que o ISC teve as piores médias. No caso das personas 3, os avaliados não conseguiram nenhum resultado, por isso a representação com média de 20 minutos, que foi o tempo total de cada teste. Essas informações estão relacionadas com a

eficiência e efetividade na utilização do software. Ou seja, pode-se dizer que as personas do tipo 1 foram as mais eficientes e o software mais efetivo foi o Scratch.

Os usuários que conseguiram um resultado mais rapidamente foram 2 personas do tipo 1 que ao utilizar o Scratch em aproximadamente apenas 1 minuto conseguiram o resultado, um fato interessante que vale a pena ser citado é que ambos já haviam tido experiências com o Scratch anteriormente. E os usuários que mais demoraram e nem chegaram a conseguir um resultado foram 2 personas do tipo 3 ao utilizar o ISC.

Entender as interações e o funcionamento das aplicações não foram os únicos responsáveis pelos resultados deste teste, um fator de muita importância é a quantidade de passos desnecessários existentes em cada software. Por exemplo, o Scratch, já poderia vir com o comando de inicio na área de trabalho, visto que alguns usuários tiveram dificuldades para encontrar o mesmo. Enquanto o ISC solicita informações desnecessárias, como por exemplo que cada item adicionado seja atribuído a uma janela, mesmo quando existe apenas uma janela na aplicação, e a criação de variáveis para alguns blocos até se as mesmas não sejam utilizadas e por último que o projeto seja salvo para ser rodado. Já o Alice, por exemplo, solicita duas confirmações na hora de adicionar determinado objeto e poderia também haver um item na caixa onde o resultado da aplicação é mostrado que permitisse configurar a mesma para que seja fechada automaticamente quando finalizada a animação.

Quadro 4.9 - Tempo médio em minutos para cada avaliado conseguir o primeiro resultado

	Persona 1	Persona 2	Persona 3	Total
Scratch	2	4	4	3,3
Alice	4	5	8	5,6
ISC	4	12	20	12
Total	3,3	7	10,6	**7**

Em relação à nota que os avaliados atribuíram aos softwares por suas funcionalidades, as personas 1 foram as que atribuíram as maiores notas, e o Scratch foi o software que mais agradou os usuários. Essas informações, mostradas no Quadro 4.10, estão relacionadas com a satisfação dos usuários ao utilizar os softwares, ou seja, se os mesmos se sentiram a vontade, e a utilização foi agradável.

Quadro 4.10 - Nota média atribuída pelos usuários aos softwares

	Persona 1	Persona 2	Persona 3	Total
Scratch	8,4	7,9	7,5	7,9
Alice	7,9	4,2	4,2	5,4
ISC	5,9	2,3	0,7	3,0
Total	7,7	4,8	4,1	**5,4**

Para a nota de usabilidade ou facilidade de uso, conforme o Quadro 4.11, o comportamento foi parecido com o anterior, contudo nesse quesito as personas do tipo 2 foram as que atribuíram as menores notas no geral. Esse comportamento provavelmente se deve ao costume em utilizar softwares mais intuitivos que os referidos neste trabalho.

Quadro 4.11 - Nota média atribuída pelos usuários à facilidade de uso dos softwares

	Persona 1	Persona 2	Persona 3	Total
Scratch	8,8	6,0	8,5	7,8
Alice	7,3	2,7	2,2	4,1
ISC	6,0	1,0	0,5	2,5
Total	7,4	3,2	3,7	**4,8**

Ao serem questionados a cerca do sentimento em relação à utilização dos softwares os avaliados responderam conforme o Quadro 4.12 que mostra a

quantidade de usuários que elogiaram ou criticaram os sistemas. Curiosamente mesmo após dar notas baixas para o software e para sua usabilidade alguns usuários responderam como satisfeitos nesse quesito.

Quadro 4.12 - Classificação dos usuários em satisfeitos ou frustrados

Scratch	Persona 1	Persona 2	Persona 3
Satisfeito	6	4	5
Frustrado	0	2	1
Alice	Persona 1	Persona 2	Persona 3
Satisfeito	6	4	2
Frustrado	0	2	4
ISC	Persona 1	Persona 2	Persona 3
Satisfeito	4	2	0
Frustrado	2	4	6

Além dos diversos dados quantitativos, os testes de usabilidade também proporcionaram diversos dados qualitativos da utilização dos sistemas, esse dados foram coletados juntamente com os demais, porem não existe uma maneira precisa de medi-los. Eles foram observados durante os testes e registrados pelo avaliador, para exemplificar alguns serão usados ilustrações e frases ou gestos que os avaliados apresentaram durante os testes. Esses dados serão expostos a seguir.

Devido ao fato de os sistemas serem de certa forma simples, com poucas telas, praticamente todos os usuários utilizaram os mesmos da forma esperada, as personas do tipo 1, sempre buscaram utilizar alguma interação a mais, através do uso de laços para repetição de comandos por exemplo. Já as personas do tipo 2 se preocuparam com a estética do resultado final, editando manualmente os objetos pré-definidos nos programas por exemplo. As personas do tipo 3 ficaram localizadas em um meio termo entre os dois outros tipos.

Os maioria dos usuários não teve dificuldades em explorar a maior parte das funcionalidades dos softwares, o que apresentou mais problemas nesse sentido foi o Alice devido à grande quantidade de informações e configurações disponíveis simultaneamente.

Durante os testes, não ocorreu falhas de comunicação entre os sistemas e os usuários, ou seja, os usuários conseguiram executar as tarefas sem se perder, e conscientes dos passos seguintes.

Os usuários conseguiram compreender a grande maioria dos elementos da interface do Scratch, já no que diz respeito ao Alice e ISC houve dificuldades em alguns pontos. No Alice, alguns usuários tiveram dificuldades para entender deveriam arrastar os itens dos menus para a área de trabalho. Em relação ao ISC, devido aos rótulos dos blocos não apresentarem uma linguagem natural mais objetiva, as personas dos tipos 2 e 3 sabiam que os blocos tinham alguma funcionalidade mas não exatamente qual era.

4.6.1 Frases dos usuários ao utilizar os sistemas

Ao utilizar os sistemas diversas frases foram coletadas pelo avaliador, as mais expressivas estão listadas a seguir, separadas por cada aplicativo.

Frases sobre o Scratch:

- “O que é um Sprite?” Persona do tipo 3, antes de clicar em adicionar um novo Sprite.
- “Eu encontrei um comando legal mas agora não consigo mais achar“ Persona do tipo 1, ao navegar entre o menu de tipos de blocos.
- “Que legal, dá pra desenhar“ Persona do tipo 2 após clicar no botão de “pintar novo objeto”.
- “Parece o Paint“ Persona do tipo 2 durante a utilização da funcionalidade de pintar novo objeto.

- “Quero redimensionar o objeto mas não acontece nada“ Persona tipo 1, ao tentar redimensionar um objeto clicando no botão de “crescer objeto“
- “Não tem a tecla de aplicar“ Persona do tipo 1, antes de descobrir o botão de início.
- “parece um fluxograma“ Persona do tipo 2, logo quando completou a interação de conectar os blocos.
- “onde fica a linha do tempo?“ Persona do tipo 2, após adicionar diversos blocos e apresentar dificuldades para planejar o tempo correto de cada interação.
- “o delete não funciona“ Persona do tipo 1 ao tentar apagar uma série de blocos usando a tecla *delete* do teclado.
- “o comando de passos é na verdade de movimento“ Persona 1 ao utilizar o bloco de efetuar passos.

Frases sobre o Alice:

- “O que é *instance*?“ Persona 3 depois de clicar para adicionar um novo objeto.
- “não sei onde ele foi parar“ Persona 3 após adicionar um novo objeto.
- “massa, é 3D“ Persona 1 após adicionar um novo objeto.
- “não custava nada o delete funcionar“ Persona 1 após tentar utilizar a tecla delete.
- “não sei qual é a posição inicial“ Persona 1 ao tentar configurar um movimento.
- “esse botão de adicionar objetos está sem visibilidade, ele podia ficar no topo“ Persona 1 depois de clicar no botão de adicionar objeto.

- “ele tem tantas opções que eu não sei nem por onde começar“ Persona 2 ao tentar configurar um novo objeto.
- “ele é todo em inglês?“ Persona 3, ao iniciar a utilização do software.
- “o scroll não dá zoom“ ao tentar utilizar o scroll do mouse para aumentar e diminuir o zoom.

Frases sobre o ISC:

- “o delete não deleta“ Persona 1 ao tentar utilizar a tecla *delete.*
- “não consigo mover os objetos usando as setas“ Persona 1 ao tentar fazer ajustes finos na posição de um objeto utilizando as setas direcionais do teclado.
- “não consigo entender o que ele está pedindo” Persona 2 após adicionar um objeto e tentar rodar a aplicação sem configurar corretamente os requisitos.
- “pra que serve uma variável do tipo *dictionary*?“ Persona 1 durante a criação de uma nova variável.
- “tem poucos eventos disponíveis“ Persona 1 ao tentar adicionar eventos diferentes na aplicação.
- “esse modelo de ligação permite uma organização melhor“ Persona 2 ao organizar seus objetos na área de trabalho.
- “não sei pra que serve, não consigo nem imaginar“ Persona 3 ao iniciar a utilização do aplicativo.
- “não consigo fazer nada com isto“ Persona 3 após alguns minutos de utilização do software.
- “essa interface é confusa, nada faz sentido“ Persona 2 após alguns minutos de utilização do aplicativo.

4.7 Análise dos resultados

Os três softwares tiveram resultados bem expressivos, a seguir serão expostos as melhores e piores características de cada aplicativo.

O Scratch teve os melhores resultados no geral, seu ponto forte claramente é a facilidade de uso, ele possui uma interface simples porém completa, seus blocos com formatos de quebra cabeça que indicam em quais podem ser associados ajudam bastante os usuários leigos a começar a reparar suas interações. Como aspectos negativos podem ser citados a ausência de comandos padrões como Crtl+z, delete, etc.

As principais possibilidade de melhorias observadas são as seguintes: adição de uma *timeline* que permita aos usuários uma melhor orientação na hora de preparar suas interações; utilizar a padronização de comandos Ctrl+z por exemplo; o bloco de iniciar poderia ser alocado no inicio da área de trabalho por padrão; alguns poucos comandos de exemplos também poderiam ser alocados por padrão, dessa forma os usuários saberiam rapidamente o funcionamento do aplicativo; menus de ajuda em português; melhorar a indicação dos blocos operadores e sensores, os quais tem um comportamento diferente dos demais; alguma seta indicando a possibilidade de arrastar os comandos pra área de trabalho; botão iniciar animação com mais destaque e com formato de *play*; melhorar a ferramenta de zoom, solicitando que o objeto seja selecionado antes da mesma.

O Alice obteve resultados intermediários, seus pontos fortes são a possibilidade de criar interações 3D; a grande quantidade de configurações e opções disponíveis de personalização e a grande quantidade de comandos pré-definidos. Como aspecto negativo pode ser citado a forma caótica que as informações são apresentadas aos usuários em algumas partes da aplicação.

O grande desafio para o Alice é simplificar as informações disponíveis simultaneamente para os usuários de forma que o aplicativo não tenha funcionalidades comprometidas. Os rótulos dos comandos e configurações precisam ser melhorados, são utilizados muitos termos técnicos, que devem ser substituídos por linguagem natural. A implementação de comandos como o delete e Ctrl+z também são indicadas. O mecanismo de zoom e navegação precisa ser melhorados, dependendo do tamanho e complexidade da interação o usuário fica limitado devido ao formato atual de controles. A adição de uma *timeline* é necessária pelos mesmo motivos do Scratch, ela permitiria que os usuários não se perdessem no tempo das interações. A forma de adicionar um objeto precisa ser refinada, caso o usuário queria apenas adicionar um objeto simples não é necessário fornecer tantas informações e possibilidades de configurações do mesmo. Uma forma de diminuir a complexidade da interface seria manter apenas uma área de trabalho como no Scratch, juntando tanto os eventos quanto os métodos em uma só região.

O ISC foi o software com os piores resultados, a abordagem de interligar os blocos agradou bem poucos usuários. Seu ponto forte é a possibilidade de portar sua aplicação para diversas plataformas como HTML5 ou Android. E Como ponto negativo pode ser citado a dificuldade em adicionar simples funcionalidades na interação, as exigências de configurações são demasiadas mesmo para ações bem simples.

Como melhorias para o ISC podem ser citadas: reestruturação dos botões do menu superior, que apesar de pouco usados tem grande destaque na interface; a funcionalidade de adicionar variáveis precisa ganhar mais destaque, os ícones de adicionar e excluir nem são mostrados como clicáveis até o mouse ser posicionado sobre os mesmos, um ótimo exemplo de péssimo *affordance*; outro item com menos relevância do que deveria é o menu *drop-down* de seleção do tipo de bloco, que passou até despercebido por alguns usuários; os títulos dos blocos precisam ser mais intuitivos, utilizando linguagem natural; utilização de comandos padrão como o Ctrl+z por exemplo; as ligações e

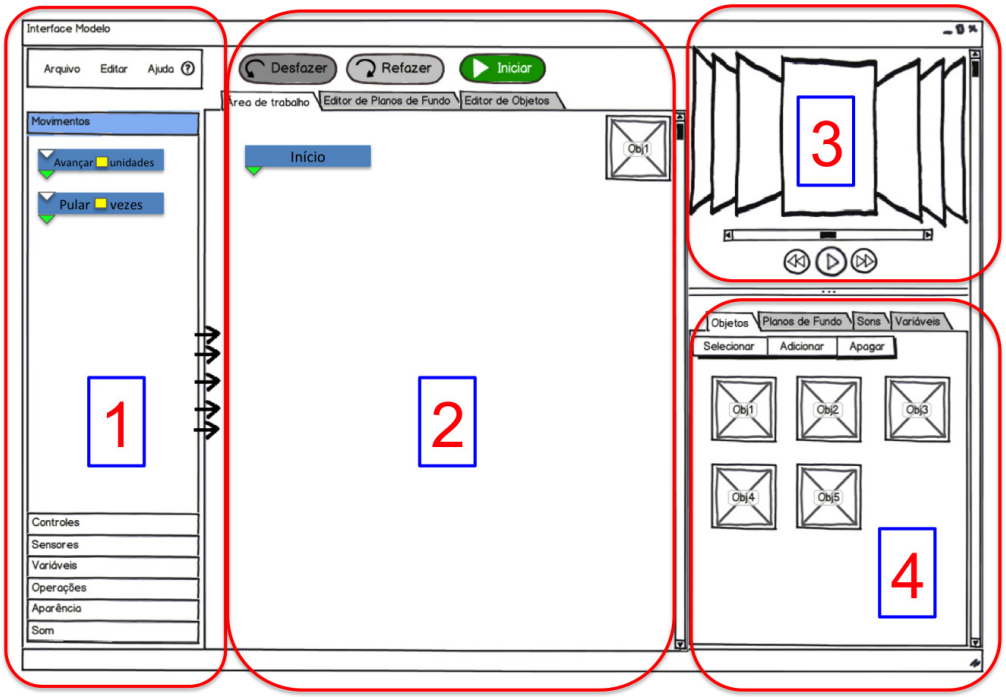
conexões dos blocos deveriam ter mais destaque, alguns usuários tiveram dificuldades de conecta-las devido ao seu tamanho diminuto; maior simplicidade para adicionar bloco, sem tantas exigências de configurações; possibilidade de zoom na área de trabalho tanto através do *scroll* quanto por botões na interface, para projetos grandes esse item é essencial; trocar o botão *run* por iniciar; traduzir os menus de ajuda e a interface para português.

5 PROPOSTA DE MODELO

O modelo foi preparado com a utilização do software Balsamiq Mockups, que é uma ferramenta de prototipação. Como era de se esperar o modelo se assemelha com o Scratch, afinal este foi o que obteve os melhores resultados nos testes de usabilidade.

Como se pode observar na Figura 5.1, a organização do software se divide em 4 áreas. Na lateral esquerda está localizada a área 1 com o menu *drop-down* de blocos de comandos, a área central, chamada de área 2, foi organizada através de abas de conteúdo, já na lateral direita superior existe agora uma *timeline,* área número 3, que também tem a função de pré-visualização dos resultados e na lateral direita inferior está localizado o menu de adição e exclusão dos itens a serem utilizados na aplicação, chamada de área 4.

Figura 5.1 - Tela inicial do modelo



Na área 1 está localizado o menu de funcionalidades do aplicativo, com o formato padrão de arquivo, editar e ajuda, presente na grande maioria dos softwares desktop atuais. Além disso essa área contem o menu *drop-down* de seleção de blocos de comando, a escolha por este tipo de menu se deu pela facilidade de uso do mesmo, deixando claro para os usuários que todos os sub-menus possuem características semelhantes e que o motivo da existência dessa divisão é apenas por uma questão de melhora na agilidade de busca por determinado bloco.

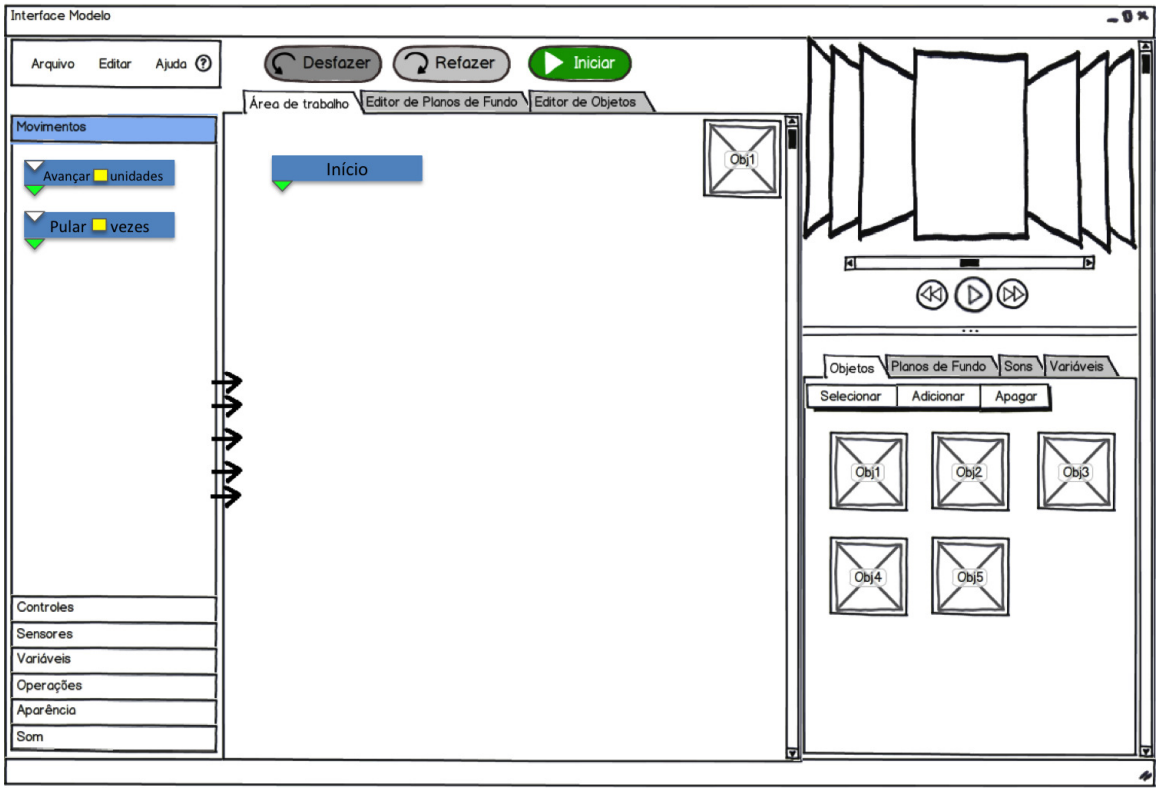
Na área 2 estão localizados os botões de controle de edição e o botão que inicia a aplicação, além disso, foram centralizadas em abas as janelas desse que é menu principal do aplicativo. As abas são: área de trabalho, que é o local onde serão montados os aplicativos utilizando os blocos disponíveis na área 1, nessa aba o objeto selecionado será representado por uma miniatura dele mesmo no canto superior direito da aba; em seguida estão as abas de edição de plano de fundo e edição de objetos, que tem por função possibilitar a edição desses itens.

A área 3 contém a *timeline* ou linha do tempo do aplicativo e seus comandos, nessa linha do tempo cada quadro de tempo do aplicativo é representado por uma tela que será substituída pela seguinte mais a direita formando o *preview* da saída do software. Nela poderão ser feitas pequenas edições tais como mover um objeto ou plano de fundo utilizando o mouse. No caso de softwares que não tenham uma organização cronológica essa linha do tempo pode apresentar as telas da saída, permitindo uma fácil visualização, edição e organização das mesmas.

Na área 4 está localizado o quarto e último menu do aplicativo, ele é organizado também utilizando a padronização no formato de abas para navegação entre suas funcionalidades, são elas objetos, planos de fundo, sons e variáveis. Cada aba dessas possui uma lista dos seus respectivos itens e permite selecionar para a área 2 ou adicionar ou ainda apagá-los. Uma

funcionalidade dessa região é no caso de um usuário desejar manter mais de uma área de trabalho com dois ou mais itens abertos simultaneamente na área 2, basta clicar com o botão direito no item escolhido e selecionar a opção de abrir em nova aba. Essa funcionalidade também fica acessível no menu editar do aplicativo, e se o usuário desejar pode inclusive configurar para cada item novo já ser aberto em uma nova aba. Essa funcionalidade não teve mais ênfase na criação do modelo pois já pode ser considerada como avançada e causadora de perdas de comunicação em usuários iniciantes.

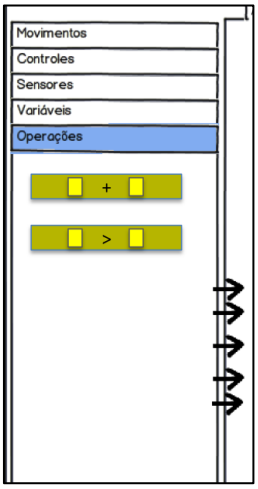
Figura 5.2 - Tela de controles



Todo o sistema foi montado em português, como pode ser observado na Figura 5.2, pois seu foco seria atender primeiramente os usuários dele aqui no Brasil. Os botões de controle: iniciar, desfazer e refazer foram bem centralizados, permitindo o fácil acesso em qualquer momento da utilização dos mesmos.

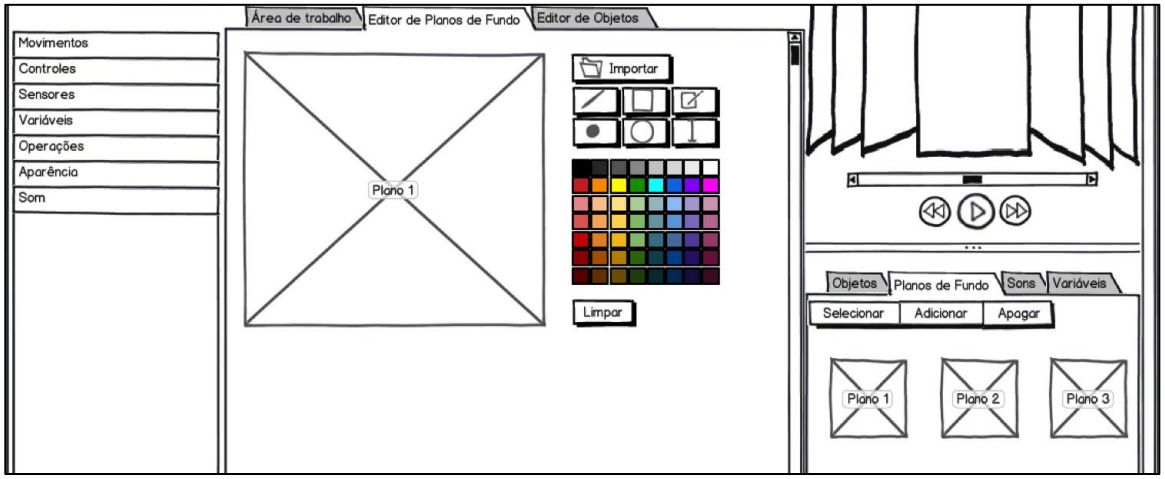
Os blocos de comandos foram padronizados da seguinte forma: blocos amarelos representam os locais onde alguma informação é necessária, essa informação pode ser algum numeral, alguma frase, ou algum outro bloco de cor amarela como pode ser observado na Figura 5.3.

Figura 5.3 - Tela de operações



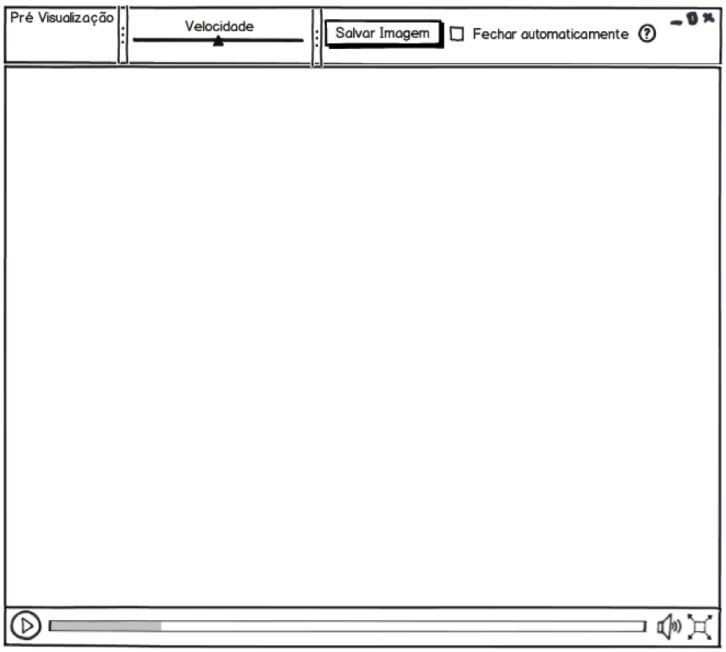
As setas servem para influenciar ainda mais os usuários na hora de entender o funcionamento do aplicativo, essas setas poderiam desaparecer conforme o uso e experiência aumentarem.

Figura 5.4 - Tela de edição de planos de fundo



Na Figura 5.4 é mostrada a tela de edição de vídeo com seus comandos básicos e configuração diferente da área de trabalho, onde o plano de fundo que é o foco dessa operação recebe mais atenção. Quando esta aba estiver selecionada, tanto as setas indicativas quanto o menu *drop-down* serão desativados de forma a impossibilitar que os seus usuários tentem utilizar essas funcionalidade em locais sem necessidade ou por engano

Figura 5.5 - Tela de visualização dos resultados



A Figura 5.5 ilustra a interface de visualização completa, as características da mesma são: possibilidade de modificar o tempo das ações, aumentando ou diminuindo a velocidade, a possibilidade de capturar imagens do resultado, um *checkbox* verificando o interesse em configurar esta janela para fechar automaticamente e por fim a interface padrão de controle de vídeos, que possibilita iniciar, pausar, controlar o som e o tamanho da tela.

6 CONCLUSÃO

O estudo das interações humano-computador e da usabilidade dos softwares são de grande importância para o desenvolvimento de aplicações de sucesso em um mercado que esta cada vez mais concorrido.

Para avaliar a qualidade dessas interações os testes de usabilidade cumprem seu papel possibilitando aos desenvolvedores e designers tirar grandes conclusões a respeito dos sistemas avaliados, além de proporcionar ao mesmo tempo ideias e soluções para os possíveis problemas detectados.

A aplicação desses testes nos três diferentes softwares de autoria permitiu fazer uma comparação eficiente entre os mesmos, na qual foram identificados diversos problemas de usabilidade em cada um como por exemplo quantidade de erros durante a realização de determinada tarefa, e também seus pontos fortes, partes intuitivas das interfaces. Todos esses dados foram coletados com a utilização de entrevistas, formulários e dos testes de usabilidade em si.

A organização dessas informações permitiu que fosse criada uma base de dados que foi utilizada para a construção do modelo para os softwares de autoria automatizada de conteúdo.

O resultado deste trabalho foi positivo e serve para mostrar que as questões de usabilidade devem ser levadas em consideração no desenvolvimento de qualquer sistema. A satisfação e a produtividade dos usuários aumenta e, dessa forma, o software e seus desenvolvedores passam a ser vistos como provedores de um serviço de maior qualidade. Além disso, também possibilitou a criação do modelo que pode servir como base para o planejamento do desenvolvimento de projetos de softwares dessa área.

Como trabalhos futuros pode-se citar o desenvolvimento de um software de autoria utilizando o modelo de interface construído neste estudo, e também a

realização de testes de usabilidade em outras classes de softwares, como softwares de edição de imagem, edição de vídeo, IDEs, etc.

7 REFERÊNCIAS

Aho, A.V.; Sethi, R.; Ullman, J.D. Compiladores: princípios, técnicas e ferramentas. Rio de Janeiro/RJ: LTC, 1995.

Alice. Sponsors. Disponível em: http://alice.org/index.php?page=sponsors/sponsors, 2012b. Acesso em 15/08/2012.

Alice. What is Alice? Disponível em http://alice.org/index.php?page=what_is_alice/what_is_alice, 2012a. Acesso em 15/08/2012.

Apple Inc. The App Store. Disponível em http://www.apple.com/iphone/built-in-apps/app-store.html. 2012. Acesso em 05/08/2012.

Avizienis, A., et al., Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing 1; Los Alamitos, CA: IEEE Computer Society, 2004.

Barbosa, Simone D. J.; Silva, Bruno S. Da. Interação Humano-Computador. Rio de Janeiro. Elsevier, 2010.

Bias, R.; Mayhem, D. Cost-Justifying Usability. San Francisco, CA: Morgan Kaufmann Publishers, 2005.

Botella, Federico; Gallud, Jose A. Tesoreiro, Ricardo. Using Interaction Patterns in Heuristic Evaluation. 2011.

Brasil Padrões e-Gov: Cartilha de Usabilidade / Ministério do Planejamento, Orçamento e Gestão, Secretaria de Logística e Tecnologia da Informação - Brasília: MP, SLTI, 2010.

Castro, Thaís H. C., Sistemização da aprendizagem de programação em Grupo. Tese de Doutorado, Programa de Pós-Graduação em Informática, Pontifícia

Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2011. Disponível em http://www.maxwell.lambda.ele.puc-rio.br/18366/18366_1.PDF Acesso em 05/08/2012.

Conwa, Matthew. et al. Alice: Lessons Learned from Building a 3D System For Novices. CHI. 1999.

Cooper, A. The Inmates Are Running the Asylum: why High-Tech Products Drive Us Crazy and How to Restore Sanity. Sams Publishing, 1999.

Cooper, A.; Reinmann, R.; Cronin, D. About Face 3: The Essentials of Interaction Design. New York, NY: John Wiley & Sons, 2007.

Courage, C.; Baxter, K. Understanding your users: a practical guide to user requirements, methods, tools, and techniques. San Francisco, CA: Morgan Kaufmann Publishers, 2005.

Delamaro, M. E.; Maldonado, J.C; Jino, M. Introdução ao teste de software. Elsevier, 2007.

Edwards, Benj. The Computer Mouse Turns 40. 2008. Disponível em http://www.macworld.com/article/1137400/mouse40.html Acesso em 12/08/2012.

Ferreira, Aurélio B. de H., Novo dicionário Aurélio da língua portuguesa. São Paulo: Editora Positivo, 2009.

Ferreira, Kátia G. Teste de Usabilidade, Monografia de Final de Curso: Especialização em Informática, Universidade de Minas Gerais. Belo Horizonte, Brasil, 2002.

Gadelha, Julia. A Evolução dos Computadores. 2009. Disponível em http://www.ic.uff.br/~aconci/evolucao.html Acesso em 12/08/2012.

Gibson, J.J. The theory of affordance. In: R. Shaw e J. Bransford (eds.), Perceiving, Acting and Knowing. Hillsdale, NJ: Erlbaum, 1977.

Guimarães, Rodrigo L. Composer: um ambiente de autoria de documentos NCL para TV digital interativa. Tese de Mestrado, Programa de Pós-Graduação em Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2007.

Hix, G.; Hartson, H. Developing User Interfaces: Ensuring Usability Through Product and Process. New York, NY: John Wiley & Sons, 1993.

ISC. Intro. Disponível em http://www.radicalbreeze.com, acesso em 15/07/2012.

ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VTDs) Part 11: Guidance on Usability. ISO, 1998.

ISO/IEC 9126: Software Engineering – Product Quality. ISO, 1991.

Junior, Vital P. D. S. Software Educacional De Autoria E A Metodologia Webquest. 2011.

Kantner, Laurie. Techniques for Managing a Usability Test. IEEE TRANSACTIONS ON PROFESSIONAL COMMUNICATION, VOL. 37, NO. 3, SEPTEMBER, 1994.

Kuniavsky, M. Observing the User Experience: A Practitioner's Guide to User Research. San Francisco, CA: Morgan Kaufmann Publishers, 2003.

Lima, Bruno S., Azevedo, Roberto G., Moreno, Marcelo F., Soares, Luiz Fernando G. Composer 3: Ambiente de autoria extensível, adaptável e multiplataforma. 2010. Disponível em http://www.telemidia.puc-rio.br/sites/telemidia.puc-rio.br/files/2010_10_blima.pdf Acesso em 11/08/2012.

Lindgaard, Gitte. Usability Testing and System Evaluation. A guide for designing useful computer systems. New York: Chapman & Hall, 1994.

Lunduke, Bryan, Illumination Software Creator Beta 1. Disponível em http://lunduke.com/?p=1141, 2010. Acesso em 16/08/2012.

Maloney, J. et al., Scratch: A Sneak Preview. Second International Conference on Creating, Connecting, and Collaborating through Computing. Kyoto, Japan. 2004.

Maloney, John, et al. Programming by Choice: Urban Youth Learning Programming with Scratch. SIGCSE'08, Portland, Oregon, USA. 2008.

Maloney, John, et al. Scratch: A Sneak Preview. Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing. IEEE Computer Society Washington, DC, USA. 2004a.

Mathew, Alex R., Abri, Ahmed A. Human-Computer Interaction (HCI): An Overview. 2011.

Menezes, P. B.; Diverio, T. A. Teoria da Computação. Sagra-Luzatto, 2000.

Nielsen, J. Enhancing the explanatory power os usability heuristics. Proceedings os ACM CHI'94. 1994.

Nielsen, J.; Molich, R. Heuristice valuation ofusers interfaces. Proceedings os ACM CHI'90. 1990.

Nielsen, Jakob B. Interative User-Interface Design. Computer, IEEE. 1993.

Nielsen, Jakob. Let's Ask the Users: Tools, techiniques, and concepts to optimise user interfaces. SunSoft, Mountain View, CA. IEEE, 1997.

Norman, D. A. Psychology of Everyday Things. Basic Books, 1988.

Perry, M. Distributed Cognition. In: J. M. Carrol (ed.), HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science. San Francisco, CA: Morgan Kaufmann, 2003.

Prates, R.O.; Barbosa, S.D.J. Avaliação de Interfaces de Usuário – Conceitos e Métodos. Jornadas de Atualização em Informática, XXIII Congresso da SBC, 2003.

Prates, R.O.; Barbosa, S.D.J. Introdução à Teoria e Prática da Interação Humano Computador fundamentada na Engenharia Semiótica. Em: T. Kowaltowski & K. Breitman (orgs.) Atualizações em informática 2007 XXVII Congresso da Sociedade Brasileira de Computação, JAI/SBC, 2007.

Presman, R.S. Engenharia de Software, Quinta Edição. Rio de Janeiro: McGraw-Hill, 2002.

Pressman, Roger S. Engenharia de Software. 3 ed. São Paulo: Makron Books, 1995.

Pruitt, J.; Adlin, T. The Persona Lifecycle: keeping people in mind through out product design. San Francisco, CA: Morgan Kaufmann Publishers, 2006.

Resnick, M., Kafai, Y., Maeda, J. ITR: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers. Proposal [funded] to the National Science Foundation, Washington, DC. 2003.

Roddy, Brian, Markowitz, Sidney, Wang-Epelman, Hernán. User Interfaces for Authoring Systems with Object Stores. IEEE Proceedings of COMPCON. 1996.

Roddy, Brian; Markowitz, Sidney; Wang, Hernán E. User Interfaces for Authoring Systems with Objects Stores. IEEE Proceedings of COMPCON. 1996.

Rosenbaum, Stephanie, Walters, Dennis R. Design Requirements For Reference Documentation Usability Testing. 1988.

Rubin, J. & Chisnell, D. Handbook Of Usability Testing: How Plan, Design, and Conduct Effective Tests, 2ª edição. Indianapolis, IN: Wiley Publishing, Inc., 2008.

Rubin, J. Handbook of Usability Testing. New York, NY: John Wiley& Sons, 1994.

Santos, Sérgio L., Teixeira, F.G. Design de uma Interface de Interação Tridimensional com Foco na Usabilidade e no Desempenho Gráfico. PgDesign. Porto Alegre, 2010.

Scratch. About Scratch. Disponível em http://info.scratch.mit.edu/About_Scratch, 2011. Acesso em 15/08/2012.

Sharp, H.; Rogers, Y.; Preece, J. Interaction design: beyond human-computer interaction, 2ª edição. New York, NY: John Wiley & Sons, 2007.

Souza, C. S. da.; et al. The Semiotic Inspection Method. Anais do VII Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais, IHC. 2006.

Souza, C.S. de.; Leitão, C.F. Semiotic Engineering Methods for Science Research in HCI. Em: J.M. Carrol (ed.) Synthesis Lectures on Human-Centered Informatics. Princeton, NJ: Morgan & Claypool Publishers, 2009.

Souza, C.S. The Semiotic Engineering of Human-Computer Interaction. Cambridge, MA: The MIT Press, 2005.

Stallings, William. Arquitetura e Organização de Computadores – Projeto para o Desempenho. Quinta Edição. Prentice Hall. São Paulo. 2003.

Teixeira, Mário A. M. et al., Projeto Pedagógico do Curso De Ciência Da Computação. Disponível em http://www.deinf.ufma.br/cocom/files/arquivos/ProjPedagogico_CP_2007_final.pdf Acesso em 05/08/2012.

Tognazzini, B. How user testing saves Money, Ask Tog. Disponível em http://www.asktog.com/columns/037TestOrElse.html, 2000. acesso em 20/08/2012.

Turing, A. M., On Computable Numbers, With An Application To The Entscheidungs problem, 1936. Disponível em http://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf Acesso em 05/08/2012.

Wang, Qiong. Usability Research of Interaction Design for E-commerce Website. 2011. IEEE.

Wharton, C. et al. The Cognite Walk throught Method: A Prectitioner's Guide. In: R. Mack & J. Nielsen (eds.) Usability Inspection Methods. New York, NY: John Wiley& Sons. 1994.