Serviço Nacional de Aprendizagem Comercial
Faculdade Senac Porto Alegre
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

ANDERSON AUGUSTO ARMANI

PLANO DE TRABALHO

SIMULADOR DE ITERAÇÕES HUMANAS PARA TESTES DE SOFTWARE

Porto Alegre
2014

ANDERSON AUGUSTO ARMANI
PLANO DE TRABALHO

SIMULADOR DE ITERAÇÕES HUMANAS PARA TESTES DE SOFTWARE

Plano de Trabalho, apresentado como requisito parcial à obtenção da aprovação do projeto de TCC1 do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, pela Faculdade Senac Porto Alegre.

Orientador: Prof. Me. Luciano Zanuz

Porto Alegre
2014

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ABES	Associação Brasileira das Empresas de Software
ATP	Associação dos Transportadores de Passageiros de Porto Alegre
IDC	International Data Corporation
IDE	Integrated Development Environment (Ambiente Integrado para Desenvolvimento de Software)
SGBD	Sistema Gerenciador de Banco de Dados
TRI	Transporte Integrado de Porto Alegre
UML	Unified Modeling Language (linguagem unificada de modelagem)

SUMÁRIO

1 APRESENTAÇÃO GERAL DO PROJETO

A Industria Brasileira de Software e Serviços vem obtendo crescimento considerável nos últimos anos. O setor cresceu 26,7% em 2012 de acordo com estudos realizados pelo International Data Corporation (IDC) em parceria com a Associação Brasileira das Empresas de Software (ABES). Para 2014 é esperado um crescimento de pelo menos dois dígitos. Estes números impulsionam ainda mais a preocupação com a Qualidade de Software.

Entende-se por Qualidade de Software a “Capacidade de um produto de software de satisfazer necessidades explicitas e implícitas quando utilizado sob condições especificas”, segundo a norma ISO25000 (ISO25000, 2008). Logo, há a necessidade de medir o nível de satisfação de um software para poder atestar sua qualidade. Ao processo de aferição da qualidade de software da-se o nome de Teste de Software.

Teste de Software é "... o processo de executar um programa com o objetivo de encontrar erros." (MYERS, 1979). Este processo pode ser dividido em duas técnicas de teste: Testes Funcionais ou caixa-preta e Testes Estruturais ou caixa-branca. Os testes estruturais analisam o código fonte afim de garantir que este é estruturalmente sólido e que funcione no contexto técnico onde está instalado, já os testes funcionais tratam o código fonte como uma caixa preta se atendo aos requisitos com o objetivo de analisar o comportamento do software desenvolvido.

Quando há a necessidade de testes funcionais repetitivos ou que serão utilizados periodicamente, utiliza-se testes automatizados. Este tipo de teste utiliza uma ferramenta de teste que gera entradas de dados para o software a ser testado, analisando as saídas geradas pelo mesmo. Uma das principais ferramentas de testes automatizados é o software de testes [1]*Selenium*, ele caracteriza-se por simulações de clicks de mouse, entrada de dados em campos simulando dados por teclado, análise de campos de saída, entre outros. Ou seja, simulações de iterações humanas com o software a nível digital, sem extrapolar ao mundo físico.

Alguns equipamentos não permitem o uso de softwares de automatização de testes funcionais, por não possuírem suporte, por rodar em ambientes restritos ou por rodar em ambientes com poucos recursos de hardware e software. Um exemplo de software que não permite este tipo de teste são os softwares embarcados ou softwares que utilizem interfaces físicas externas como um teclado diferenciado, comandos via rede ou serial, interface com outros dispositivos como cartões inteligentes… Pouco ouve-se falar sobre o uso da robótica em testes funcionais.

O que este projeto propõe é o desenvolvimento de um software para a automatização de testes funcionais através do controle das ações de um braço robótico. O software deverá simular iterações humanas de entrada de dados para sistemas, análise de respostas e geração de relatório de testes.

---

[1] Selenium: http://www.seleniumsoftware.com/

2 DEFINIÇÃO DO PROBLEMA

No ano de 2007 começou a operação técnica do sistema de bilhetagem eletrônica Transporte Integrado de Porto Alegre (TRI), que surgiu da necessidade de avanços na prestação de serviços e na melhoria dos sistema então existente (ATP, 2012). Este sistema surgiu da parceria entre a Associação dos Transportadores de Passageiros (ATP), a Empresa Pública de Transporte e Circulação (EPTC) e a Prodata Mobility Brasil, empresa responsável pelo desenvolvimento técnico do sistema.

> O Tri – Transporte Integrado é o sistema de bilhetagem eletrônica de Porto Alegre que consiste na arrecadação automática da passagem de ônibus, através da utilização de cartões inteligentes com créditos eletrônicos. Isso possibilita que a cidade tenha um sistema mais evoluído de transporte público coletivo, como é utilizado em outros grandes centros do Brasil e do mundo. O sistema de bilhetagem eletrônica também tem como objetivo integrar itinerários e beneficiar os passageiros, através de descontos na tarifa para quem utiliza mais de uma linha e para quem utiliza outros meios de transporte coletivo (SITE TRI, 2014).

O TRI, nos ônibus de Porto Alegre, opera com validadores de passagem eletrônica da linha da empresa Prodata Mobility Brasil. Toda a ação de um usuário, seja ele um passageiro ou um operador do sistema, com o validador é através do uso de cartão inteligente sem contato.

Segundo (ATP, 2012) em 2007 o sistema começou com o cadastramento de aproximadamente 100 mil idosos, em 2012, já com todos os perfis de usuários contemplados, ele operava com aproximadamente 227 mil isenções: idosos, portadores de necessidades especiais e pessoas de baixa renda; 240 mil estudantes e 600 mil cartões de valor como vale-transporte e passe antecipado, cartões pré-pago.

Com o crescimento do sistema, cresceram também as preocupações com possíveis problemas que podem afetar os usuários, consequentemente os envolvidos em manter o projeto TRI buscaram qualificar o desenvolvimento do sistema e métodos para melhor aferir a qualidade dos softwares gerados. Segundo o setor de Projetos da ATP, em 2012 foi desenvolvido um método de aferir a qualidade do software gerado para os validadores de passagem, fora designada uma equipe para testes e gerado um Script de Testes capaz de simular os mais diversos usos de cartões nos validadores. Assim o sistema cresceu em confiabilidade, passou a apresentar menos erros aos usuários e melhorou o tempo de transação de cartões, segundo dados internos da própria ATP.

Com o intuito qualificar ainda mais os processos de desenvolvimento do TRI o setor de Projetos iniciou em 2013 o contato com a empresa Zero-Defect, empresa especializada em testes de software e validação de sistemas. A empresa analisou os processos de validação do desenvolvimento do TRI e montou uma proposta de desenvolvimento de software baseado em testes, com a utilização de “robos” de testes ou testes automatizados. Porém algo que chamou atenção da ATP foi a falta de uma solução automatizada para os testes funcionais dos validadores de passagem. Este tipo de testes se mostrou necessário pois na proposta técnica da Zero-Defect a maneira encontrada para testar os validadores continuava muito próxima a atual, com operações realizadas manualmente, mudando somente a forma de validar os dados obtidos, que passariam a ser validados através de comparação com resultados pré-definidos.

O método atual de testes, apesar de ter se mostrado eficaz no passado, por sua extensão e quantidade de recursos utilizados se tornou um dos principais agravantes de tempo

entre o desenvolvimento de uma nova versão de software dos validadores e a sua real implantação no sistema. Os testes até a presente data, março de 2014, demandam cerca de 3 dias de trabalho e uso de até 3 funcionários da ATP, um analista de testes, um testador e um analista de suporte e implantação.

Haveria no mercado alguma forma de automatizar a execução destes script de testes da ATP?

Atualmente robôs já são utilizados em testes de equipamentos, um exemplo é o Messina RS. “O sistema de automação de testes baseado em robô Messina RS é uma solução flexível e eficiente para a verificação de eletrônicos de interiores e intretenimento” (BERNER-MATTER.COM, 2013). Este equipamento pode simular sequências de operações humanas afim de realizar teste funcional de equipamentos. Ele é empregado atualmente para testes de sistemas de interiores de veículos como rádios de veículos da marca Ford.

O Messina RS é um grande exemplo do uso de robôs em testes, porém poucos são os exemplos do uso de robôs para testes de software e equipamentos em geral.

O mercado de robôs para uso em automatização de testes é promissor, mas carece de uma solução de caráter genérico, que se adapte a inúmeros casos de testes, e de custo acessível, capaz de fomentar o uso de robôs em teste funcionais. Um robô com essas características poderá atingir uma grande gama de empresas que possuem hoje testes funcionais realizados manualmente, atenderia também as necessidades da ATP e do projeto TRI de automatizar seus testes funcionais em busca de maior confiabilidade e dinamismo.

3 OBJETIVOS

Este capítulo apresenta os objetivos deste trabalho, divididos em geral e específicos, estes são apresentados nas sessões a seguir.

3.1 OBJETIVO GERAL

O objetivo geral deste projeto é o desenvolvimento de um software para a automatização de testes funcionais através do controle das ações de um braço robótico. O software deverá simular iterações humanas de entrada de dados para sistemas, analisar respostas e gerar relatório de testes.

3.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

a) controle de um braço mecânico;
b) interface de cadastro de objetos;
c) gerenciador de configurações do sistema;
d) tratamento de retorno de status de teste via serial;
e) cadastro de Ações;
f) cadastro de Testes;
g) motor de execução de Teste;
h) relatório de Teste;
i) cadastro de Script de teste;
j) motor de execução de Script;
k) relatório de Script;
l) controle de usuários e permissões;
m) validar o sistema através da implantação nos testes do projeto TRI.

4 ANÁLISE DE TECNOLOGIAS/FERRAMENTAS

A seguir são listadas as tecnologias e ferramentas que serão utilizadas no desenvolvimento deste trabalho. As escolhas estão baseadas em critérios técnicos como o uso em múltiplas plataformas, robustez, disseminação e aceitação no mercado de software, e critérios econômicos como o uso de sistemas [2]*Open Source*, ou seja, sem custos de licença para uso.

4.1 LINGUAGEM JAVA

Java é uma linguagem de programação orientada a objetos robusta que foi "…testado, refinado, estendido e comprovado por uma comunidade dedicada de desenvolvedores, arquitetos e entusiastas[…]" (JAVA.COM). É uma linguagem que permite o desenvolvimento para multiplas plataformas. Encontramos java de sistemas Desktop, em múltiplos sistemas operacionais, datacenters, supercomputadores, celulares até produtos domésticos como aparelhos de TV.

É a segunda linguagem mais utilizada no mundo, segundo ranking da (TIOBE.COM, 2014) em março de 2014.

Java é a linguagem de programação mais adequada a este projeto devido as características:

a) software livre, não gera custos pela sua utilização;

b) multiplataforma, softwares desenvolvidos em Java, quando bem programados, podem ser compilados para vários sistemas operacionais;

c) documentação, por ser muito utilizada, apresenta uma documentação vasta e acessível;

d) reusabilidade, por ter muitos componentes já desenvolvidos permite maior agilidade no desenvolvimento de soluções.

4.3 ECLIPSE

Eclipse é uma IDE Open Source para desenvolvimento de softwares Java originalmente desenvolvida pela IBM em novembro de 2001 e mantida atualmente pela comunidade Eclipse Foundation. Esta IDE suporta outras linguagens de programação como C/C++, PHP, Python, Scala… além de ser utilizada para desenvolvimento de aplicações para sistema Android. Sua construção é modular permitindo customizações de acordo com as necessidades do projeto ou do desenvolvedor. Por ser Open Source e por possuir módulos de integração com o repositório de versionamento de código GitHub foi escolhida para uso neste projeto.

---

[2] Open Source: Software de licença livre para uso e alterações.

4.4 POSTGRESQL

PostgresSQL é um Sistema Gerenciador de Banco de Dados (SGBD) relacional Open Source, que agrupa, armazena, manipula e recupera dados em forma de tabelas. Banco de Dados Relacional é um conjunto de tabelas que contém os dados de um sistema qualquer. PostgreSQL é um sistema que permite o gerenciamento de banco de dados, definição de estruturas e definição de regras de acesso aos dados.

Cada tabela pode ser dividida em linhas e colunas. Um apontamento de linha e coluna define um campo. Suas linhas, ou tuplas, são formadas por uma lista ordenada de colunas, que representam os registros.

O relacionamento entre as tabelas ocorre através de seus campos chave. Um campo chave pode ser um ou mais campos que determinam a unicidade de cada registro.

PostgreSQL é hoje um dos principais SGBD, e há algum tempo goza de grande admiração. "PostgreSQL é o mais avançado servidor de banco de dados Open Source" (MOMJIAN, 2000).

PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de ativo desenvolvimento e uma arquitetura que tem provado sua forte reputação de confiabilidade, integridade e correção de dados (POSTGRESQL.ORG).

Por suas características de software livre, sua confiabilidade, robustez e interoperabilidade entre sistemas operacionais, será utilizado neste trabalho.

4.5 PGADMIN

O pgAdmin é a plataforma de administração e desenvolvimento PostgreSQL Open Source mais popular e mais rica em recursos (PGADMIN.ORG). Por isso será utilizado neste trabalho no desenvolvimento do banco de dados e posteriormente na sua administração.

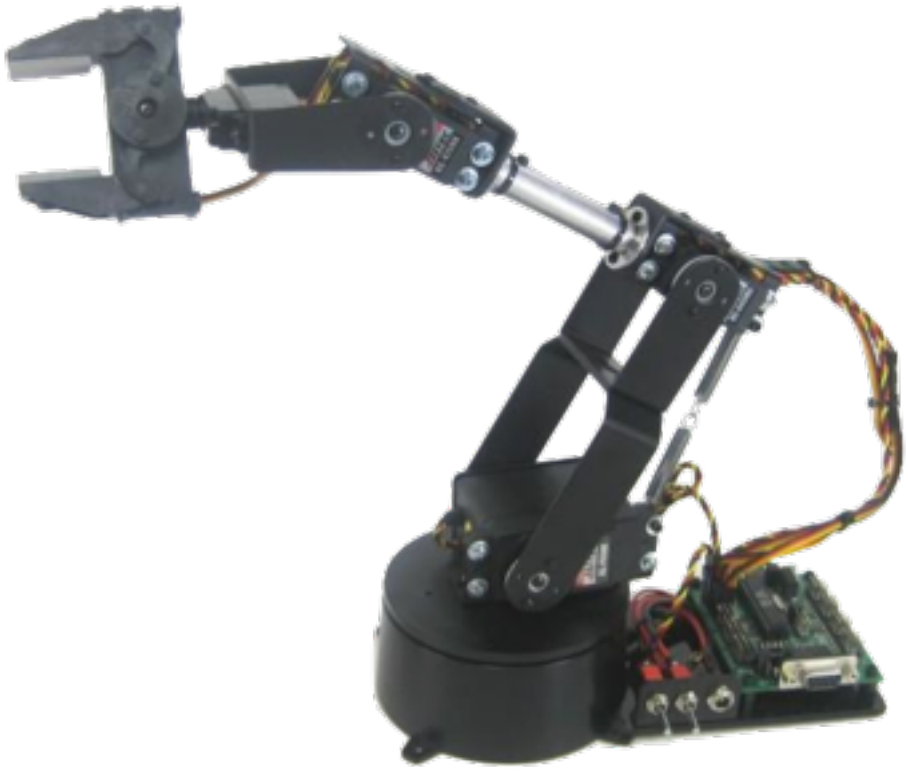
4.6 KIT ROBÓTICO AL5B

O AL5B é um braço robótico capaz de produzir movimentos rápidos, precisos e repetitivos. Possui base com capacidade de rotacionar, ombro de plano único, cotovelo e movimento de punho, uma garra funcional, e opcionalmente, com capacidade de rotação do pulso (LYNXMOTION, 2014). Figura 1: Kit AL5B da Lynxmotion, apresenta uma imagem do kit montado. O kit utilizado neste trabalho não possui a capacidade de rotacionar o pulso. A integração entre o AL5B e o software de controle se dá através de comunicação via porta serial [3]RS232 do computador, em caso de computadores sem porta serial deverá ser utilizado cabo conversor usb para serial RS232.

---

[3] RS232: é um padrão de comunicação de rede entre dispositivos muito utilizado em computadores mais antigos.

Figura 1 - Kit AL5B da Lynxmotion



Fonte: http://www.lynxmotion.com/c-126-al5b.aspx

4.7 TRELLO

É uma ferramenta de gerenciamento de projetos baseado no paradigma Kanban. Como este paradigma o Trello utiliza um quadro de atividades dividido em colunas e linhas. As colunas representam o estado das atividades e as linhas o fluxo de uma atividade. As atividades são representadas por cartões ou quadros coloridos, que podem variar em cor e/ou tamanho para demonstrar sua prioridade.

Este sistema é colaborativo e pode ser compartilhado por uma equipe de trabalho para acompanhamento do fluxo de trabalho, indicação das atividades a serem executadas e estado de cada atividade. Quando bem utilizado este paradigma pode acelerar entregas, possibilitar melhor visualização das atividades e da produtividade da equipe de trabalho e o melhor balanceamento das atividades.

Por este trabalho ser desenvolvido individualmente, o Trello será utilizado como meio de controle de tarefas e [4]*sprints*. Através dele serão controladas as atividades a serem realizadas por sprint, as atividades em desenvolvimento e as atividades concluídas.

---

[4] Sprint: Período de atividades segundo a abordagem de desenvolvimento Scrum.

4.8 ASTAH COMMUNITY

Astah é uma ferramenta de software para o desenvolvimento em UML que auxilia no desenvolvimento de diagramas como: Caso de Uso, Classes, ER… Será utilizado no desenvolvimento de diagramas para a análise e desenvolvimento deste projeto por ter boa usabilidade, atender as necessidades de projeto e por ter seu uso livre de cobranças.

4.9 GITHUB

O GitHub é um servidor de versionamento de código fonte online. Ele é colaborativo, permite gerenciamento e revisão de código por vários usuários. Por ser um repositório online o GitHub permite acesso de diversos lugares o que evita a necessidade de estar logado em uma rede privada específica, como outros versionadores.

Será utilizado neste trabalho pela sua praticidade e por ser livre de custos no modelo de repositório aberto.

5 DESCRIÇÃO DA SOLUÇÃO

Este capítulo apresenta a solução proposta para o desenvolvimento do Simulador de Iterações Humanas para Teste de Software, a integração entre as partes envolvidas na operação, na figura 2, e as definições de hierarquia entre os tipos de ações do sistema.

Figura 2 - Integração das partes envolvidas no teste

Fonte: O próprio Autor

Será desenvolvido um Software em java que controlará o braço robótico AL5B permitindo ações que simulem interações humanas com equipamentos. Essas iterações serão direcionadas a testes funcionais do equipamento alvo.

Teste funcional avalia o comportamento do software a ser testado, para tal, fornece dados de entrada, realiza uma determinada ação, obtém um resultado e o compara com um resultado esperado previamente conhecido. Caso o resultado esperado for igual ao obtido o teste foi positivo, caso contrário o teste foi negativo. O teste funcional avalia o software como uma caixa preta, sem saber como o software foi desenvolvido, sendo importante somente a saída gerada conforme a entrada disponibilizada.

A figura 2 apresenta a iteração entre as partes do simulador. O usuário do simulador cadastrará ações do braço robótico, a posição dos objetos a serem utilizados no testes e a posição do equipamento a ser testado, agrupará essas ações formando testes e com os testes poderá gerar scripts de testes. Esses dados serão guardados em banco de dados PostgreSQL.

O software terá uma interface via RS232 com o kit AL5B por onde enviará os comandos para o kit. Esses comandos deverão gerenciar a aquisição de objetos e a utilização destes no equipamento a ser testado.

Os resultados dos testes poderão ser validados pelo simples fato da ocorrência do teste, não aguardando retorno, ou através da comparação de retorno via porta serial RS232, comparando com um resultado esperado pré-cadastrado nos testes.

O software permitirá ao usuário a definição de algumas posições para braço robótico. O usuário será capaz de agrupar essas posições do braço robótico afim de definir uma sequência de passos que gere uma Ação do braço, por exemplo:

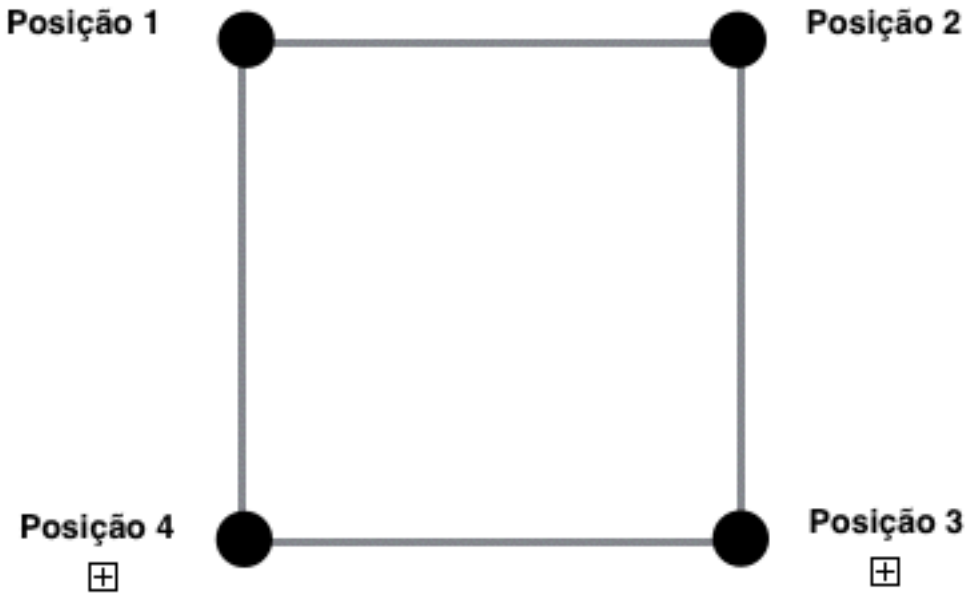
Posição 1 - Posicionar o braço mecânico sobre uma caneta;
Posição 2 - Fechar pinça do braço para pegar a caneta;
Posição 3 - Posicionar a caneta em pé sobre o canto esquerdo inferior de uma folha de papel;
Posição 4 - Arrastar a caneta até o canto direito superior da folha de papel
Posição 5 - Abrir a pinça para soltar a caneta.

Neste caso a Ação seria desenhar uma diagonal partindo do canto inferior esquerdo até o canto superior direito de uma folha de papel posicionada dentro do raio de ação do braço mecânico.

O software permitirá também que o usuário agrupe algumas ações gerando um teste, como mostra a Figura 3 - Exemplo de multiplas ações, que consiste no agrupamento de inúmeras ações a serem executados seguindo sua ordem de cadastro.

Ação 1 - Desenhar um reta horizontal na posição 1 até a posição 2
Ação 2 - Desenhar um reta vertical da posição 2 até a posição 3
Ação 3 - Desenhar um reta horizontal na posição 3 até a posição 4
Ação 4 - Desenhar um reta horizontal na posição 4 até a posição 1

Figura 3 - Exemplo de multiplas ações



Fonte: O próprio Autor

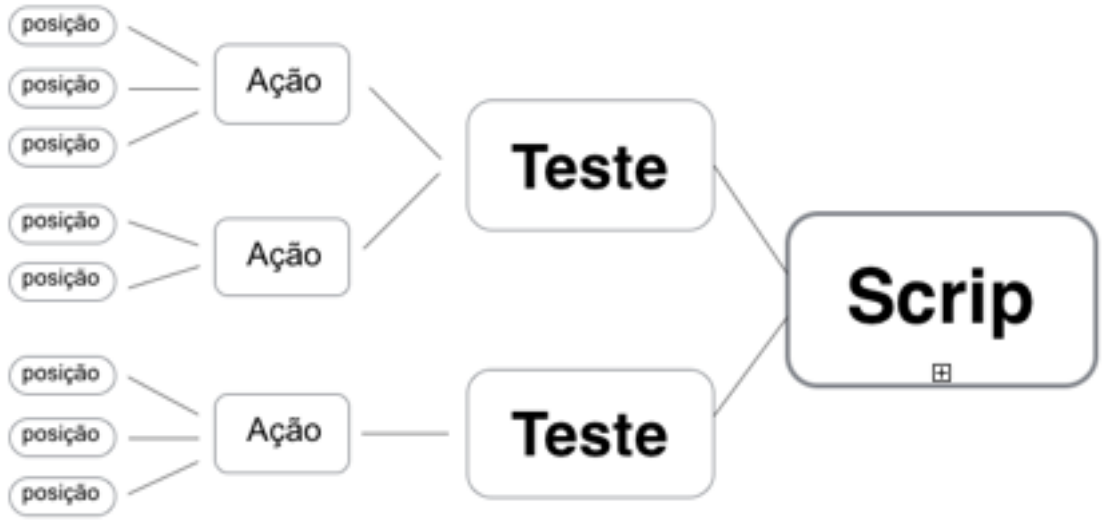
Este é um exemplo do Teste para desenhar um retângulo, que consiste no agrupamento de várias ações pré cadastradas.

O Teste será a menor ação que o software irá executar perante comandos do usuário, as Posições serão somente cadastradas para compor as Ações e estas para compor os Testes.

O maior nível de execução de testes será o Script de Testes que, seguindo a ordem, consiste no agrupamento de inúmeros Testes. Os Scripts poderão ser executados individualmente podendo o usuário escolher se quer executar o Script completo ou quais Testes que compõem o script serão realizados.

Os níveis e agrupamentos dos tipos de dados cadastrados pelo usuário está representado na Figura 4 - Ordem de grandeza dos tipos definidos. A figura demonstra que Scrips contém Testes, estes por sua fez contém uma ou mais ações. Já as ações são constituídas de posições.

Figura 4 - Ordem de grandeza dos tipos definidos



Fonte: O próprio Autor

Ao executar um Teste ou Script o software coletará dados da execução apresentando ao final um relatório de status dos testes executados. Estes dados podem ser uma simples confirmação de execução de um teste ou o status enviado pelo equipamento via porta serial, que será comparado com o esperado pelo teste gerando então o status final do teste.

O Simulador conterá ainda um sistema de usuários e permissões onde, usuários Administradores do sistema poderão criar e executar testes e usuários comuns somente poderão executar testes previamente criados. A diferenciação entre os usuários será através de login e senha.

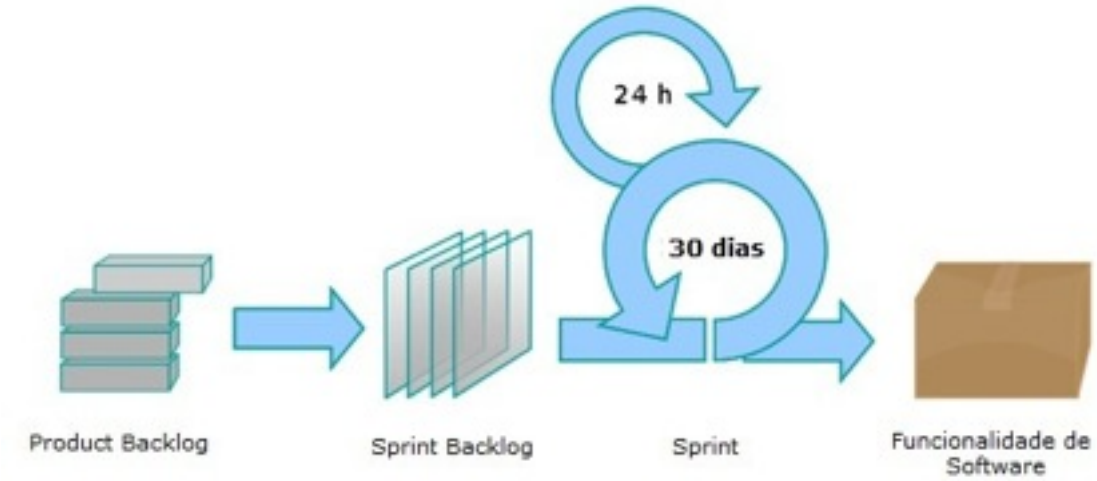
6 ABORDAGEM DE DESENVOLVIMENTO

Este trabalho utilizará, como abordagem de desenvolvimento, o método iterativo incremental baseado nas boas práticas do Scrum, adaptadas a uma equipe de apenas um desenvolvedor.

No modelo iterativo incremental as tarefas são quebradas em pequenas porções para que seu desenvolvimento seja rápido, assim as funcionalidades são incrementadas a cada novo ciclo de desenvolvimento.

O Scrum é um método de trabalho para equipes pequenas e tempos curtos de desenvolvimento de projeto (BROD, 2013, p. 49). Como mostra a Figura 5 - Visão geral do Scrum, este método trabalha com a idéia de Sprints, que são períodos curtos de tempo, tipicamente 2 a 4 semanas, onde é realizado o desenvolvimento de alguma funcionalidade do projeto. A lista de funcionalidades a serem desenvolvida é chamado de Product Backlog. Um ítem de Product Backlog irá gerar tarefas que serão armazenadas no Sprint Backlog, que é a lista de tarefas a serem desenvolvidas no Sprint.

Figura 5 - Visão geral do Scrum



Fonte: GROFFE

> A principal razão de se dividir as entregas de um projeto em sprints é justamente a questão de manter-se o controle sobre as surpresas. Dentro do período do sprint, uma parte do produto será projetada, codificada, testada e entregue ao cliente. Neste período não serão admitidas mudanças de requisitos, pois isso ampliaria o tempo de desenvolvimento. Ao final do sprint, porém, podem ser revistos os requisitos[…] (BROD, 2013, p. 50).

No scrum existem três papéis:

a) Product Owner - Patrocinador do projeto, entidade que patrocina o projeto;

b) Scrum Master - Coordenador geral do projeto, membro da equipe que garante o bom andamento do projeto;

c) Equipe - É quem realiza o desenvolvimento do projeto.

Este projeto utilizará do scrum o Product Backlog, onde serão descritas as atividades que deverão ser desenvolvidas.

O desenvolvimento será dividido em etapas de entregas ou sprints. Cada sprint terá duração de 2 semanas. Ao início de uma nova fase de desenvolvimento será definido o Sprint Backlog, que consiste em quais funcionalidades do Product Backlog serão desenvolvidas, quais suas etapas e qual a prioridade de desenvolvimento.

Durante cada sprint será realizada a análise, desenvolvimento e validação de cada funcionalidade a ser desenvolvida, gerando como documentação os diagramas listados no item 7 deste documento, Arquitetura do Sistema.

Ao final de cada sprint será gerada uma entrega, que conterá as funcionalidades definidas no Product Backlog. Esta entrega irá gerar um mínimo produto viável que será validado, como definido no item 8 deste documento.

O desenvolvimento baseado em Scrum tornará o projeto mais ágil e maleável a quaisquer alterações que se mostrem necessárias. E permitirá que ocorram entregas durante o desenvolvimento da solução, o que torna viável que este trabalho sejá realizado em dois semestres, gerando a cada etapa um produto viável e funcional. A idéia inicial do Product Backlog deste projeto é apresentada abaixo no Quadro 1 - Poduct Backlog.

Quadro 1 - Product Backlog

Prioridade	Item	Descrição
	1	Aceite do orientador
	2	Plano de trabalho
	12	Relatório parcial
	18	Relatório final
	3	Interface de controle do kit AL5B
	4	Cadastro de objetos
	5	Interface de configuração de Posições do braço robótico
	6	Gerenciador de configurações do sistema
	7	Tratamento de retorno de status via serial
	8	Interface de configuração de Ações
	9	Configuração de Teste

Prioridade	Item	Descrição
	10	Motor de execução de Teste
	11	Relatório de Teste
	13	Configurações de Script de Teste
	14	Motor de execução de Script de Teste
	15	Relatório de Script de Teste
	16	Controle de usuários e permissões
	17	Implantação e validação do sistema

Fonte: O próprio Autor

7 ARQUITETURA DO SISTEMA

Neste capítulo será apresentada a arquitetura do sistema, bem como os modelos de prototipação usados.

7.1 MODELAGEM FUNCIONAL

Na modelagem funcional serão levantados os requisitos funcionais e não funcionais e utilizado diagramas de casos de uso e de classes, desenvolvidos através do Astah Community.

7.2 MODELAGEM DE DADOS

Será utilizado o Diagrama Entidade Relacionamento (ER) para descrever o modelo de dados. Este modelo será gerado com o auxílio do Astah Community e sua implementação será através do pgAdmin e do SGBD Postgres.

7.3 MODELAGEM DE INTERFACE GRÁFICA

A cada Sprint que tiver desenvolvimento de tela, será gerado um protótipo da tela a ser desenvolvida. Este protótipo de tela deverá incluir todas as funcionalidades necessárias para a tela a ser criada.

SPRINT 1 - Plano de Trabalho

Este primeiro sprint consiste na definição do Plano de Trabalho do projeto que foi desenvolvido e constitui a versão inicial deste relatório.

SPRINT 2 - Desenvolvimento da Interface de Controle do KIT AL5B.

Consiste em desenvolver uma interface de comunicação serial RS232 para utilização em diversas plataformas. Essa interface de comunicação será utilizada para controlar o braço mecânico e, posteriormente, para adquirir os dados de retorno dos testes. Logo essas interface deve ser flexível para que possa ser usada para as duas finalidades. Consiste também no desenvolvimento do controle do envio e recepção de dados para o braço robótica AL5B através do uso da interface de comunicação serial 232.

**Descrever o padrão de comunicação serial 232
** Descrever como implementar isso no JAVA
OBS: Salientar os problemas com a comunicação serial no Java
- Lib da Oracle descontinuada
- RXTX com problemas no site
- Encontrada lib comm sem documentação específica
** Descrever padrão de comunicação com o braço robótica AL5B

O Sprintbacklog do Sprint 2 ficou definido como:

a) Configuração do ambiente de desenvolvimento: Baixar programas, extensões e configurá-los, instalar drivers e testá-los;

b) Análise de requisitos para desenvolvimento deste sprint;

c) Confecção do diagrama de caso de uso;

d) Confecção do diagrama de Classe;

e) Definição da validação;

f) Implementação;

g) Testes e validação.

Configuração do ambiente:

Foram instalados no equipamento de desenvolvimento os softwares:

* Eclipse IDE e sua extensão WindowsBuilder para criação de telas;
* Astah Community para o desenvolvimento de diagramas de caso de uso e de classes;
* Instalação de drivers para cabo USBSerial prolifc do tipo PL2303;

Análise dos requisitos:

A baixo segue a análise de requisitos para este sprint, onde são listados os requisitos funcionais e não funcionais.

Requisitos Funcionais:

RF001 - Trabalhar tanto com porta Serial quanto USBSerial;

RF002 - Portabilidade entre os sistemas Windows, Linux e Mac Os;

RF003 - Usar eventos para recepção dos dados;

RF004 - Classe serial genérica para permitir uso em outras funcionalidades;

RF005 - Disponibilizar método que liste todas as portas seriais disponíveis de acordo com o sistema operacional no qual está rodando.

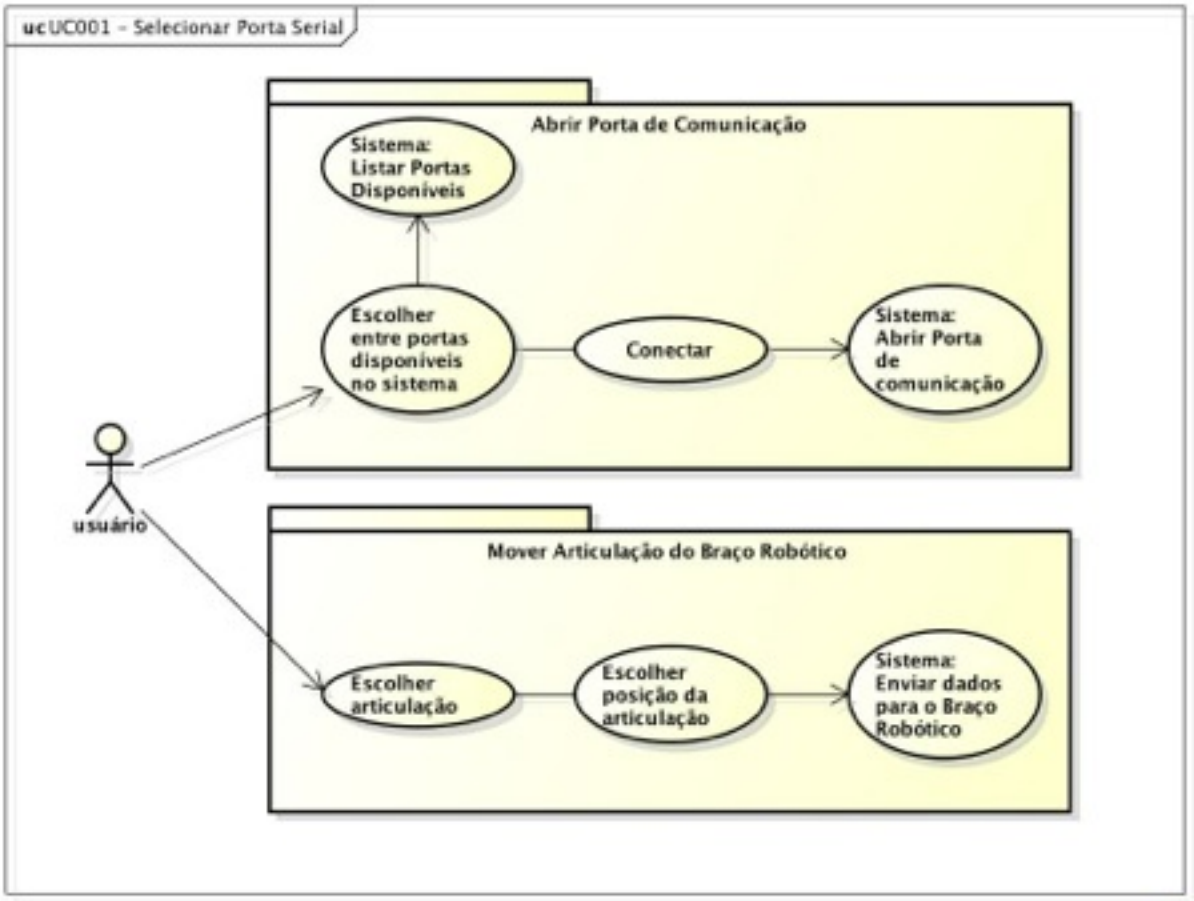
Requisitos Não Funcionais:

RNF001 - Ao ocorrer algum erro na porta a mesma deverá ser fechada.

Diagrama de Caso de Uso:

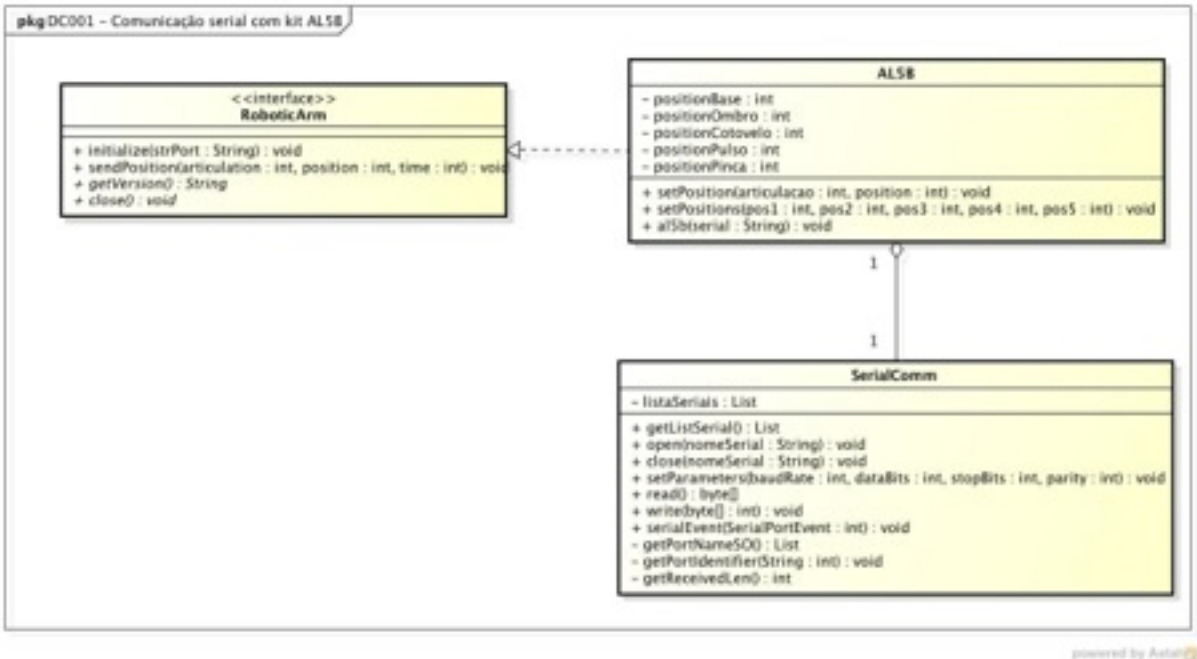
* Sistema seleciona as interfaces válidas - retorna lista de interfaces
* Usuário seleciona interface de comunicação desejada;
* Usuário abre a porta serial para comunicação já configurando-a;
* Usuário seleciona articulação que deseja mover;
* Usuário seleciona posição a ser enviada para a articulação;
* Sistema envia dados para o braço robótico.

Figura 6 - Diagrama de Casos de Uso do Sprint 1



Fonte: O próprio autor.

Diagrama de Classes:
Figura 7 - Diagrama de Classes do Sprint 1



Fonte: O próprio autor.

Definição de Validação:

A validação do sprint1 será através de testes unitários desenvolvidos para comprovar o perfeito funcionamento do sistema e também o desenvolvimento de todos os requisitos funcionais e não funcionais definidos para o sprint.

Para testar o requisito funcional RF001, o sistema será testado tanto em computador do tipo desktop com porta serial quanto em computador do tipo notebook, que não possui porta serial e deve fazer uso de cabo USBSerial. Para estes testes serão utilizados dois equipamentos um computador **Dell Vostro 2990**, onde será utilizada a porta serial, e um **Macbook MB305**, onde será utilizado um cabo USBSerial da Prolifc (PL2303).

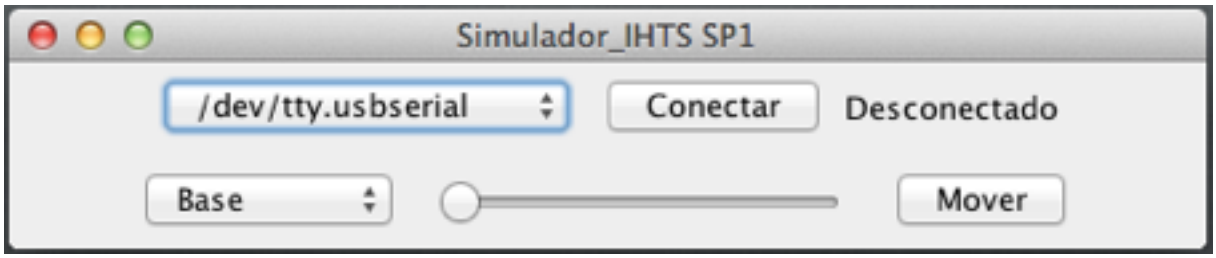
O requisito funcional RF002 também utilizará dos dois equipamentos para validar o uso em diferentes sistemas operacionais. O uso no OS X da Apple será testado no **Macbook MB305**, já os testes no windows e no linux serão realizados no **Dell Vostro 2990** que possui dois sistemas operacionais instalados windows 7 e linux Fedora 8.

A recepção de dados através de eventos será testada com o auxílio de um "curto-circuito" entre os pinos de envio (TX) e recepção (RX) da porta serial. Este curto fará com que todos os dados enviados sejam recebidos no mesmo momento. Se for programado através de eventos o sistema deverá ao final do envio ter recebido todos os bytes enviados.

Para testar o RF005 o sistema será rodado nos três sistemas operacionais (OS X, Linux e Windows) e deverá listar as portas contidas nos mesmos. Deverá ser plugado cabo USBSerial em todos os testes para garantir que liste tanto portas seriais como usbserial. No OS X deverá listar portas com o nome "/dev/tty.usbserial", no Linux "/dev/ttyUSB" e no Windows "COM".

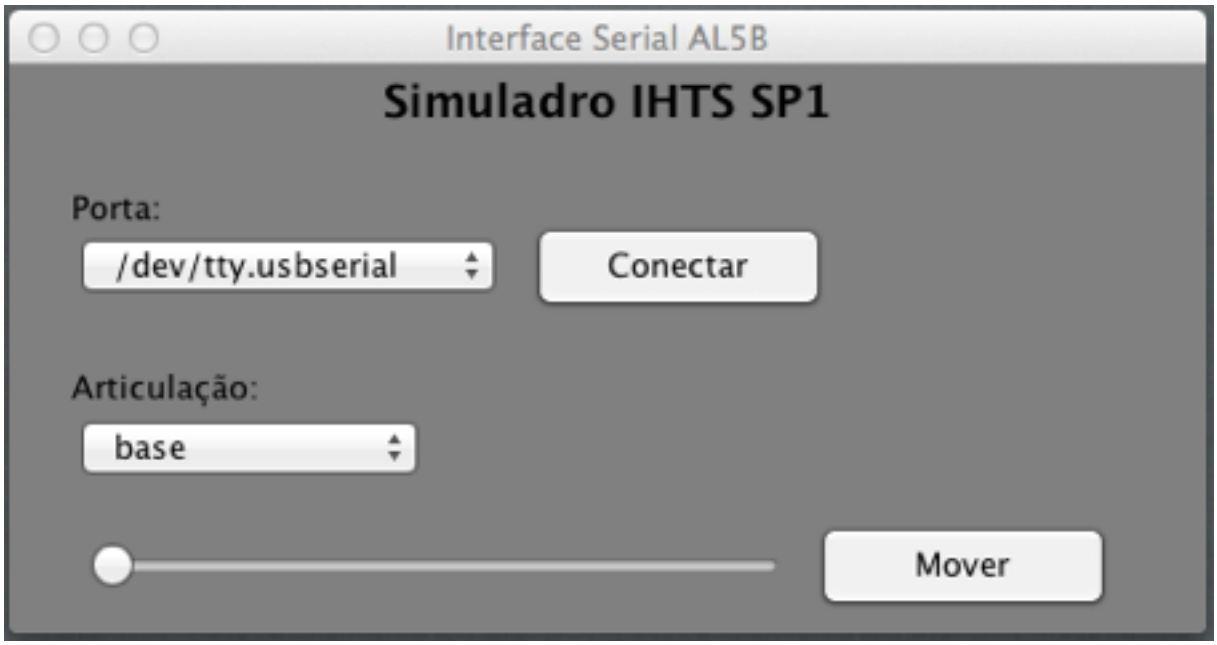
Implementação:

Figura X - Primeira versão do aplicativo de teste do sprint 1 no OS X.



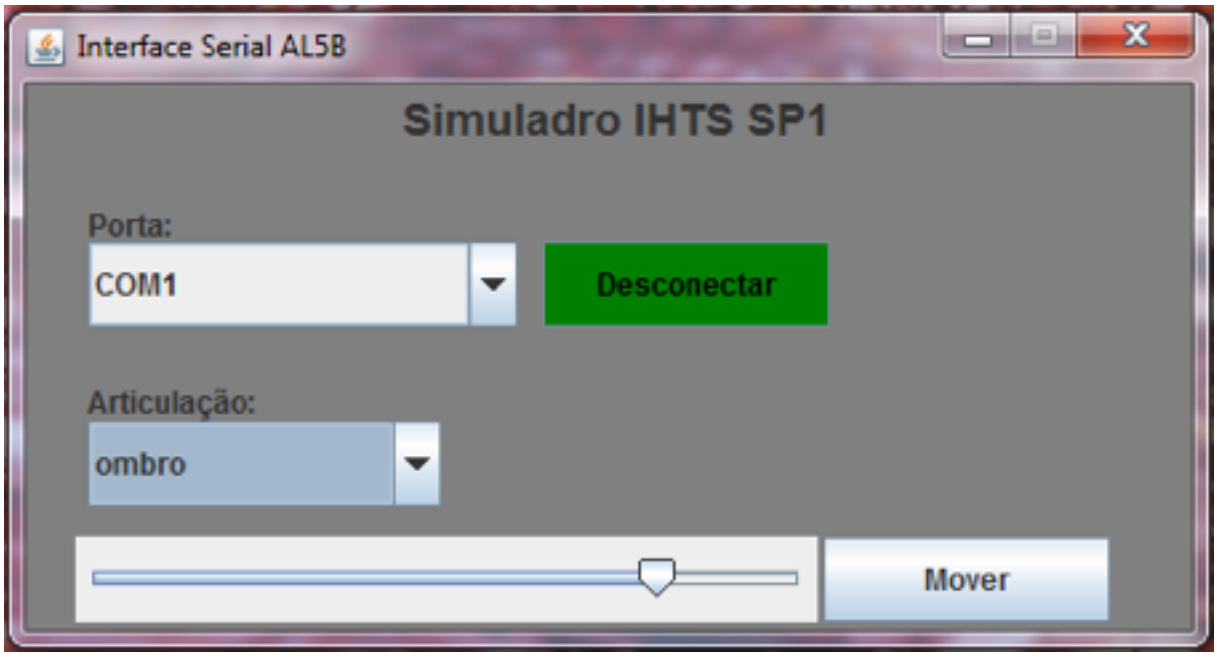
Fonte: O próprio autor.

Figura X - Segunda versão do aplicativo de teste do sprint 1 no OS X.



Fonte: O próprio autor.

Figura X - Aplicativo de teste do Sprint 1 no Windows



Fonte: O próprio autor.

Figura X - Aplicativo de teste do Sprint 1 no Linux

Fonte: O próprio autor.

Testes e Validação:

8 VALIDAÇÃO

Este item descreve como será a validação deste projeto. Ele é dividido em duas subseções Estratégia, que definirá quais os modelos de validação serão utilizados e Consolidação de Dados, que define como serão apresentados os dados obtidos.

8.1 ESTRATÉGIA

A validação do Simulador de Iterações Humanas para Testes de Software ocorrerá a cada final de Sprint através de confirmação de implementação de todas as tarefas definidas para o Sprint e de todos os requisitos funcionais e não funcionais especificados. Esta confirmação é de responsabilidade do desenvolvedor, no caso o autor deste projeto.

Também será validado através da implantação do Simulador no ambiente de testes dos validadores de passagem eletrônica do sistema TRI na sede da ATP. Após esta implantação o uso do Simulador será avaliado segundo a norma ISO25000 para qualidade de software, que avaliará a qualidade do software do simulador através de questionário a ser respondido pelos usuários.

A norma ISO25000 ou Software product Quality Requirements and Evaluation SQuaRE, descreve a maneira de medir a qualidade do produto de software em todo o seu ciclo de vida. Para isso a norma define requisitos de qualidade interna, externa e em uso.

O projeto como um todo e sua eficácia será avaliado segundo questionário para a equipe de testes e para os gerentes do TRI.

A implantação do Simulador no ambiente de testes do TRI transcorrerá da seguinte forma:

a) Apresentação do Simulador de Iterações Humanas para Testes de Software para toda a equipe técnica envolvida nos testes do TRI;
b) Treinamento técnico para uso do Simulador como Administrador e como Usuário;
c) Geração guiada dos primeiros Testes e Scripts de teste;
d) Criação do perfil de usuário Administrador TRI;
e) Aplicação de testes com usuários com poucos conhecimentos no sistema;
f) Liberação do equipamento para uso na automatização do sistema de testes;
g) Aplicação de testes com usuários já com experiência no sistema;
h) Aplicação de um questionário de acordo com o perfil do usuário.

8.2 CONSOLIDAÇÃO DE DADOS

Serão analisados os dados das pesquisas e sintetizados em forma de gráficos para provar a qualidade e usabilidade do software do Simulador. Através de pesquisa com os usuários e com os gerentes da ATP serão validados se os objetivos deste trabalho, descritos no item 3, foram alcançados. Para que um objetivo seja alcançado deverá atingir pontuação mínima necessária segundo métricas que serão definidas seguindo a norma ISO25000/ SQuaRE.

9 CRONOGRAMA

Quadro 2 - Cronograma

Atividade	Produto	Data Final	Descrição
	Aceite TCC1	17/03/2014	Entrega do formulário de aceite
Sprint 1	Plano de Trabalho	31/03/2014	Desenvolvimento e entrega do Plano de Trabalho
Sprint 2	Interface serial com kit AL5B	14/04/2014	Desenvolvido uma interface de controle de serial em java. Através desta interface foi desenvolvido a interface de controle do kit AL5B.
Sprint 3	-	28/04/2014	
Sprint 4	-	12/05/2014	
Sprint 5	-	26/05/2014	
	Relatório Parcial	02/06/2014	Entrega do relatório parcial
Sprint 6	Banca TCC1	09/06/2014	Banca Final TCC1
	Aceite TCC2	29/08/2014	Entrega do fomrmulário de aceite
Sprint 7	-	12/09/2014	
	Relatório de Projeto	22/09/2014	Entrega do Relatório de projeto atualizado
Sprint 8	-	26/09/2014	
	Seminário de Andamento	29/09/2014 a 03/10/2014	Seminário de andamento do projeto
Sprint 9	-	10/10/2014	
Sprint 10	-	24/10/2014	
Sprint 11	-	07/11/2014	
	Relatório Final	17/11/2014	Entrega do relatório final do projeto
Sprint 12	-	21/11/2014	
	Banca Final	24/11/2014 a 28/11/2014	Banca Final
	Relatório	08/12/2014	Entrega da versão final do relatório

Fonte: O próprio Autor

REFERÊNCIAS BIBLIOGRÁFICAS

ASTAH COMMUNITY, Site oficial. Disponível em < http://astah.net/editions/community>. Acesso em: 21 março 2014.

ATP, Associação dos Transportadores de Passageiros; **Tri para todos.** Porto Alegre: Uffizi Consultoria em Comunicação, 2012.

BERNER-MATTER.COM, Messina RS Robot-based Test Automation of Infotainment/ Interior Electronics; Disponível em <http://www.berner-mattner.com/en/berner-mattner-home/products/messina-rs/index.html>. Acesso em: 26 março 2014.

BROD, Cesar. **Scrum guia prático para projetos ágeis.** São Paulo: Novatec Editora Ltda, 2013.

GROFFE, Renato José. Desenvolvimento ágil com scrum: uma visão geral. Devmedia.2014. Disponível em <http://www.devmedia.com.br/desenvolvimento-agil-com-scrum-uma-visao-geral/26343>. Acessado em: 26 março 2014.

DRAKE, Joshua D; WORSLEY, John C. **Pratical PostrgreSQL.** O'Reilly Media, 2010.

JAVA.COM. Obtenha informações sobre a Tecnologia Java. Disponível em < http://java.com/pt_BR/about/>, Acesso em: 20 março 2014.

LYNXMOTION, Lynxmotion Imagine it. Build it. Control it. Disponível em < http://www.lynxmotion.com/c-126-al5b.aspx >. Acesso em:21 março 2014.

MYERS, Glenford J; **The Art of Software Testing.** John Wiley & Sons Inc, Hoboken, New Jersey 1979.

MOMJIAN, Bruce; **PostgreSQL: Introdution and Concepts.** Boston: AddisonWesley, 2000.

PGADMIN.ORG. pgAdmin PostgreSQL Tools. Disponível em < http://www.pgadmin.org/>. Acesso em: 20 março 2014.

POSTGRESQL.ORG. About PostgreSQL. Disponível em: < http://www.postgresql.org/about/>. Acesso em: 20 março 2014.

TIOBE.COM. TIOBE Index for March 2014. Disponível em <http://www.tiobe.com/>, Acesso em: 20 março 2014.

SITE TRI, O que é TRI. Disponível em <http://www.tripoa.com.br/o_que_e_tri.html>. Acesso em: 20 março 2014.