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

ANDERSON AUGUSTO ARMANI

RELATÓRIO PARCIAL

SIMULADOR DE ITERAÇÕES HUMANAS PARA TESTES DE SOFTWARE

Porto Alegre
2014

ANDERSON AUGUSTO ARMANI
RELATÓRIO PARCIAL

SIMULADOR DE ITERAÇÕES HUMANAS PARA TESTES DE SOFTWARE

Relatório Parcial, 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

RESUMO

Vivemos a revolução do software. Cada dia mais e mais pessoas utilizam sistemas para facilitar e automatizar tarefas, consultar base de dados, comunicar-se, controlar tarefas e objetos. Os softwares estão crescendo em complexidade e extrapolando o mundo digital, eles estão cada vez mais interagindo com o mundo real, com o mundo físico. Estas novas formas de interação, este aumento em complexidade, ao mesmo tempo que requerem novos paradigmas de teste, também possibilitam novas formas de encarar a validação de softwares e equipamentos. Este trabalho apresenta o desenvolvimento de um software, que ao utilizar de interação com robótica, visa simular iterações humanas a serem aplicadas na automatização de teste de softwares de equipamentos.

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 “robôs” 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 entretenimento” (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, que são apresentados nas seções 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.
n) Reduzir o tempo gasto nos testes dos validadores do sistema 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 múltiplas 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.

4.2 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.

4.3 WINDOW BUILDER

---

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

Plugin para o Eclipse que auxilia na criação de telas gráficas utilizando o padrão Swing ou SWT, tornando muito fácil a criação de Java GUI (Graphic User Interface), sem gastar muito tempo escrevendo código (http://www.eclipse.org/windowbuilder/). Este plugin permite o uso de técnicas de arrastar-soltar para a construção da interface gráfica de modo visual, ficando a cargo do plugin a construção do código fonte correspondente.

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.

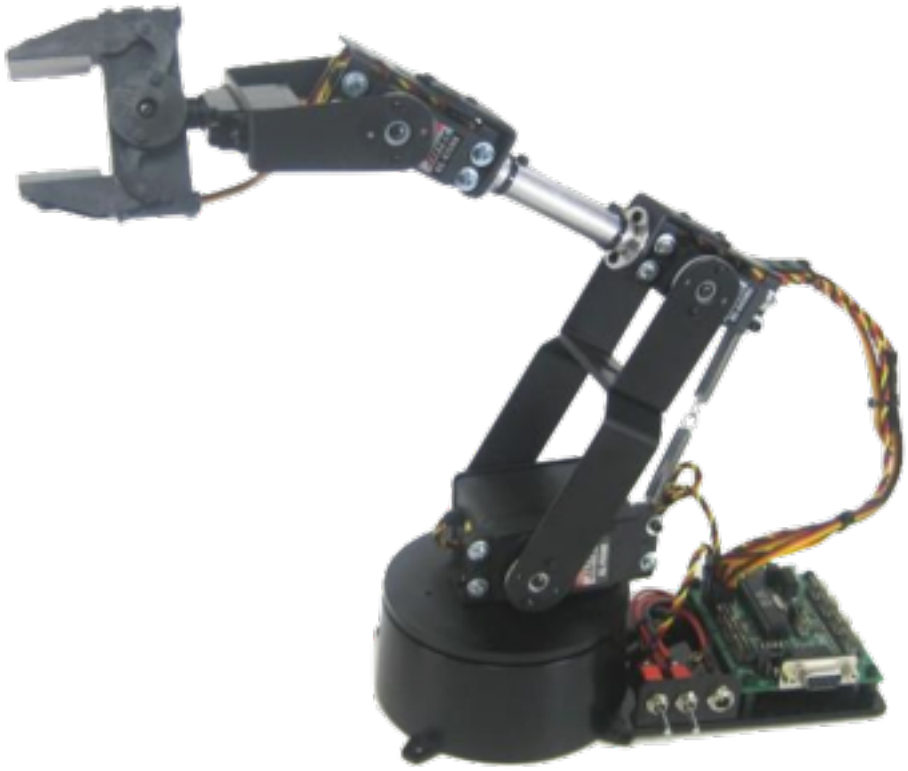
4.6 KIT ROBÓTICO AL5B

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.

4.10 DAO

Data Acess Object ou Objeto de Acesso a Dados, é um padrão de abstração de base de dados. É uma camada intermediária entre os dados e a camada de persistência, que provê operações com os dados sem expor os mesmos. Essa técnica é utilizada para isolar a camada de dados afim de obter maior controle e isolamento do código.

4.11 MVC

A sigla MVC, do nome em inglês, Model View Controller, ou Modelo Visão e Controle, é um padrão de desenvolvimento de software que visa a separação das entidades do software afim de prover robustez, reutilização e facilidade de manutenção de código.

> No paradigma MVC a entrada do usuário, o modelo do mundo externo e o retorno visual para o usuário são explicitamente separados e tratados por três tipos de objetos, cada um especializado em uma tarefa. A View gere a saída gráfica ou textual que é atribuída ao sistema. O Controller interpreta entradas de dados do usuário através do mouse e do teclado, alterando o Model e a View apropriadamente. Finalmente, o Model controla os dados de domínio da aplicação (ILLINOIS 1997).

Atualmente existem algumas variações incrementais no modelo MVC que dividem o Model em três camadas com o uso de DAO, ModelVO, ModelBO e ModelDAO. Segundo esta definição a classe VO (Value Object) serve para descrever o modelo de dados ou tabela do banco de dados. O modelo DAO especifica o acesso aos dados, no caso de banco de dados é onde estão os métodos de inserção, exclusão e alteração dos dados. O BO é a camada de de regra de negócio (Business Object) onde são realizados os tratamentos dos dados.

Neste padrão MVC de model de três camadas há uma separação ainda maior das entidades o que gera maior flexibilidade, reuno de código e facilidade de manutenção do código. Este modelo é o escolhido para ser usado neste projeto.

4.12 JAVADOC

É uma ferramenta para geração de documentação de API em formato HTML através de comentários de código.

Esta ferramenta facilita a documentação de código uma vez que ela pode ser escrita no próprio código utilizando marcações específicas. Este tipo de documentação é especialmente útil em desenvolvimento ao descrever dados de um método ou classe utilizado tais como: autor, versão, parâmetros de entrada, tipos de retornos e excessões lançadas. Contendo estes dados o desenvolvedor não precisa consultar e entender o código fonte da entidade a ser usada o que acelera o desenvolvimento.

Ao fim do desenvolvimento de um módulo ou aplicação a documentação facilita a utilização e a manutenção do mesmo evitando maiores transtornos.

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

O Software, produto deste trabalho, é desenvolvido em java e é responsável por controlar o braço robótico AL5B permitindo ações que simulem interações humanas com equipamentos. Essas iterações são 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 cadastra ações do braço robótico, a posição dos objetos a serem utilizados no testes e a posição do equipamento a ser testado, agrupa essas ações formando testes e com os testes gera scripts de testes. Esses dados são guardados em banco de dados PostgreSQL.

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

Os resultados dos testes podem 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 deve permitir ao usuário a definição de algumas posições para braço robótico. O usuário pode agrupar essas posições do braço robótico afim de definir uma sequência de passos que gere uma Ação do braço, por exemplo:

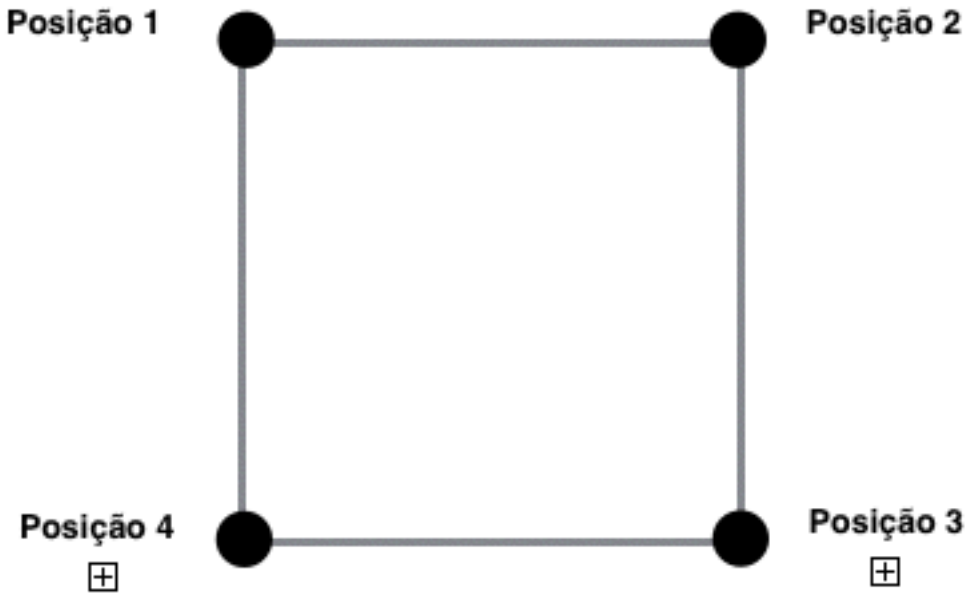
Posição 1 - Posicionar o braço mecânico sobre uma caneta;
Posição 2 - Fechar pinça do braço para pegar a caneta;
Posição 3 - Posicionar a caneta em pé sobre o canto esquerdo inferior de uma folha de papel;
Posição 4 - Arrastar a caneta até o canto direito superior da folha de papel
Posição 5 - Abrir 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 deve permitir também que o usuário agrupe algumas ações gerando um teste, como mostra a Figura 3 - Exemplo de múltiplas 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 múltiplas ações



Fonte: O próprio Autor

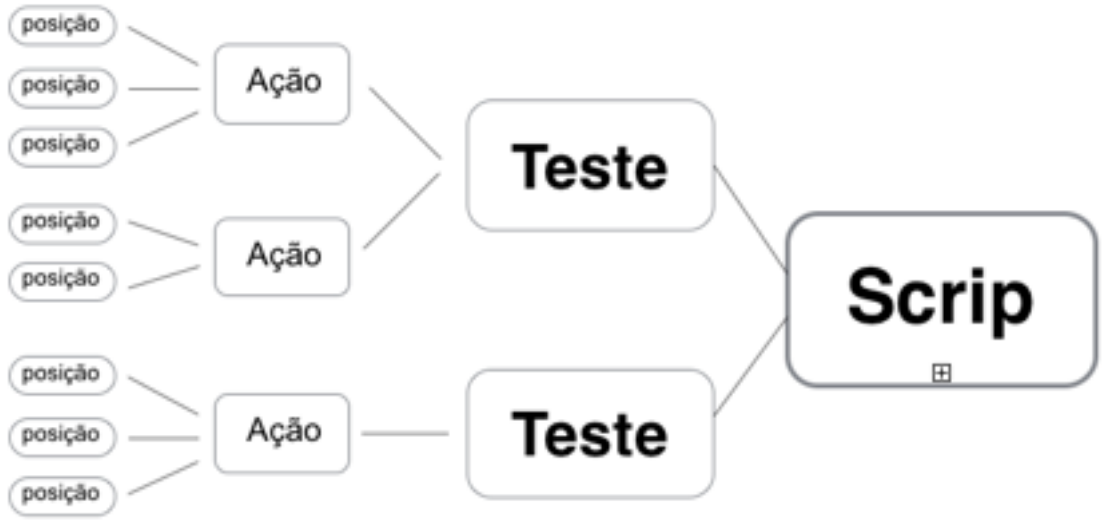
Este é um exemplo do Teste para desenhar um retângulo, que consiste no agrupamento de várias ações pré cadastradas.

O Teste é a menor ação que o software irá executar perante comandos do usuário, as Posições somente são 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 podem 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 coleta 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.

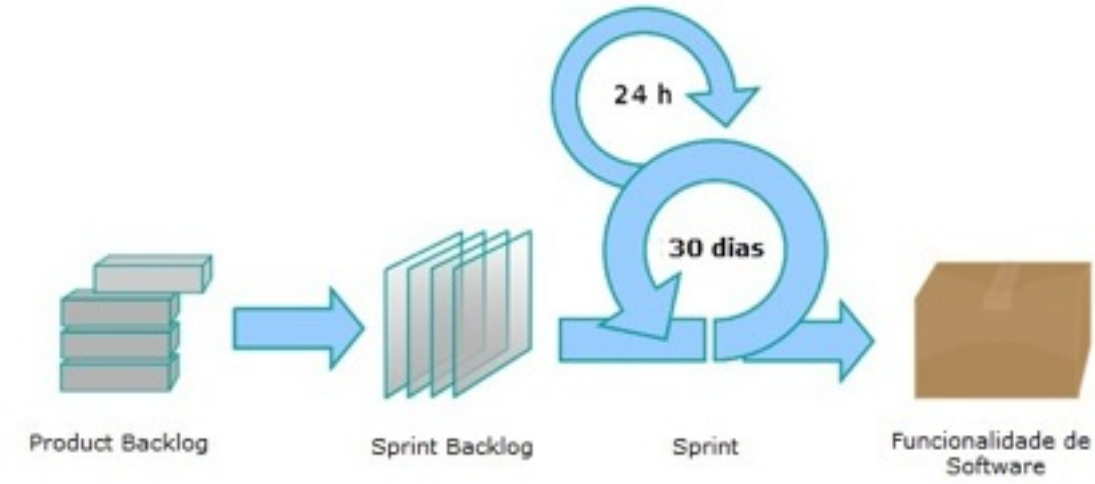
6 ABORDAGEM DE DESENVOLVIMENTO

Este trabalho utiliza, 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 item de Product Backlog, ao ser colocado para desenvolvimento, é dividido em várias tarefas que são 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 utiliza do scrum o Product Backlog, onde são descritas todas as macro funcionalidades necessárias para atender aos requisitos deste projeto.

O desenvolvimento está dividido em etapas de entregas ou sprints. Cada sprint tem duração definida de 2 semanas. Ao início de uma nova fase de desenvolvimento é definido o Sprint Backlog, que consiste em quais funcionalidades do Product Backlog são desenvolvidas na iteração em questão, quais suas etapas e qual a prioridade de desenvolvimento.

Durante cada sprint é 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 é gerada uma entrega, que conterá as funcionalidades definidas no Product Backlog. Esta entrega gera um mínimo produto viável que é então validado, como definido no item 8 deste documento.

O desenvolvimento baseado em Scrum torna o projeto mais ágil e maleável a quaisquer alterações que se mostrem necessárias. E permite que ocorram entregas durante o desenvolvimento da solução, o que torna viável que este trabalho seja realizado em dois semestres, gerando a cada etapa um produto viável e funcional.

7 ARQUITETURA DO SISTEMA

Neste capítulo, é apresentada a arquitetura do sistema, bem como os modelos de prototipação usados. Como a abordagem de desenvolvimento utilizada é iterativa incremental baseada em Scrum, alguns dos tópicos abaixo serão separados em ciclos de desenvolvimento ou sprints.

7.1 MODELAGEM FUNCIONAL

A seguir são listados o Product Backlog, o Sprint Backlog com todas as atividades, os Requisitos Funcionais e Não Funcionais e os diagramas de Casos de Uso e de Classes.

7.1.1 PRODUCT BACKLOG

No quadro 1 são listados os aspectos gerais do projeto, as funcionalidades necessárias para a construção deste e o estado atual em formato de Product Backlog. Como este é um projeto a ser desenvolvido em dois semestres, apenas parte das funcionalidades já está desenvolvida, o restante será desenvolvido a seguir, como mostra o Quadro 3 - Cronograma, no item 10 deste trabalho.

Quadro 1 - Product Backlog

Item	Descrição	Estado
Aceite do orientador	Documento de aceite do Orientador	100%
Plano de trabalho	Documento de planejamento do trabalho	100%
Relatório parcial	Relatório do desenvolvimento realizado no TCC1	100%
Relatório final	Relatório de desenvolvimento de todo o projeto	
Interface de controle do kit AL5B	Desenvolvimento de uma interface serial em Java genérica, para comunicação com o Kit AL5B e possibilidade de fácil integração com outros kits	100%

Item	Descrição	Estado
Interface de configuração de Posições do braço robótico	Desenvolvimento de uma interface gráfica que possibilite ao usuário cadastrar, listar, editar e excluir Posições do braço robótico.	100%
Gerenciador de configurações do sistema	Tela de gerenciamento de configurações iniciais do sistema: Qual porta serial está sendo usada, quais os dados de acesso ao banco de dados…	100%
Tratamento de retorno de status via serial	Tratamento de uma serial de “debug” para receber status dos testes realizados, comparar os dados recebidos com os esperados e retornar status do teste.	50% - Parte está desenvolvida na interface serial
Interface de configuração de Ações	Tela que possibilite ao usuário cadastrar, listar, editar, excluir e simular Ações	100%
Configuração de Teste	Tela que possibilite ao usuário criar, editar, excluir e simular Teste através da união de várias ações de forma ordenada	
Motor de execução de Teste	Funcionalidade que permita a execução de um Teste já cadastrado	
Relatório de Teste	Tela que mostre o status da execução de um teste	
Configurações de Script de Teste	Interface que permita ao usuário criar, editar, excluir e simular Scripts de Testes através do agrupamento ordenado de Testes	
Motor de execução de Script de Teste	Funcionalidade que permita a execução de um Script de teste já cadastrado	
Relatório de Script de Teste	Tela com resumo dos status de todos os testes pertencentes ao script	
Controle de usuários e permissões	Interface que permita a criação e edição de usuários e suas permissões	

Item	Descrição	Estado
Implantação e validação do sistema	Implantar o sistema no teste de um equipamento, realizar questionário que aponte pontos positivos e falhos do sistema. Concluir se o sistema atende os requisitos.	

Fonte: O próprio Autor

7.1.2 SPRINT BACKLOG

No quadro 2 abaixo estão descritas as atividades realizadas em cada sprint, e como, a partir do modelo incremental, foram desenvolvidos os modelos de dados e a implementação do sistema.

Quadro 2 - Sprint Backlog

Sprint	Descrição	Produto
	Documentação Inicial do Projeto	
Sprint 1	Aceite do Orientador	Documento de aceito do Orientador Assinado;
	Plano do Projeto	Plano do Projeto desenvolvido, revisado pelo orientador, impresso e encadernado.
	Configuração do ambiente	- Instalar e configurar Eclipse IDE e Plugin WindowBuilder
	Configuração do ambiente	- instalar e testar Astah Community
	Configuração do ambiente	- Instalar e testar Banco de dados Postgresql e PgAdmin
	Configuração do ambiente	- Instalar e testar Software do GitHub
	Interface de Comunicação Serial com o Braço Robótico e Configuração do Ambiente de Desenvolvimento	
	Lib Serial	Definir Lib Serial a ser utilizada no projeto
	Comunicação serial	Desenvolvimento de uma classe de comunicação serial que utilize a lib para comunicação serial
	Recepção serial por evento	Método de recepção serial por evento, para evitar perda de dados recebidos pela serial e agilizar tal processo

Sprint	Descrição	Produto
Sprint 2	Interface de Braço Robótico	Desenvolvimento de uma interface Java que descreva as principais funções de um braço robótico (RoboticArm)
	Módulo AL5B	Desenvolvimento de um módulo de comunicação com o kit AL5B utilizando a interface RoboitcArm
	Tela para teste	Desenvolver interface gráfica para software de teste
	Aplicação de teste	Desenvolver software de teste que permita: - Listar seriais do PC; - Abrir serial escolhida; - Alterar posição de uma articulação do Kit AL5b.
Sprint 3	Cadastro de Posições	
	Modelo ER	Desenvolver modelo ER para cadastro de Posições
	Tabelas no Banco de dados	Criar tabela de posições no banco de dados e testar manipulação dos dados (insert, update e delete)
	Diagrama de Classes	Desenvolver Diagrama de Classes
	Diagrama de Casos de Uso	Desenvolver Diagrama de casos de uso do sprint
	Interface Gráfica	Desenvolver interface gráfica para o cadastro de posição com: Nome, Descrição e posição das 5 articulações.
	Classes DAO	Desenvolver Classe PositionDAO que descreva a tabela de posição
	Classe Main de controle	Desenvolver Classe que controle o cadastro de Posições
Sprint 4	Estruturação MVC, Rafatoração do Código Atual e Desenvolvimento de Tela Principal do Sistema	
	Estruturação MVC	Separar o código já desenvolvido nas pastas de acordo com o sistema MVC (model, view, control)
	Refatorar Cadastro de Posições	Separar o código desenvolvido pela estrutura MVC, desenvolver as entidades faltantes.
	Tela principal do sistema	Desenvolver a tela principal do sistema com Barra de Menu, Menu de acesso rápido e painel de visualização de funções

Sprint	Descrição	Produto
	Main Controller	Desenvolver o controlador da interface principal, responsável pela inicialização do sistema e tratamento dos eventos de menu
	Inicializar Braço Robótico	Inicializar a interface serial de controle do braço robótico
	Refatorar Diagrama de Classe	Alterar o Diagrama de Classes do sistema para o padrão MVC
Sprint 5	Cadastro de Ações	
	Levantamento de Requisitos	Levantamento de requisitos para implementação do Cadastro de Ações
	Diagrama de Casos de Uso	Incrementar Diagrama de Casos de Uso com os requisitos do Sprint 5
	Diagrama ER	Incrementar Diagrama ER com as tabelas necessárias para o Cadastro de Ações
	Interface Gráfica	Desenvolver tela do cadastro de Ações Desenvolver painel de Posição de uma Ação
	Model	Desenvolver classes de model para cadastro de Ações e Lista de Posições (Action VO, BO, DAO) (ActionList VO, BO, DAO)
	Controller	Desenvolver Classe de controle para o cadastro de Ações
	Configuração do sistema	Desenvolver o controle das configurações do sitsema através de arquivo de properties.
	Tela de configuração do sistema	Tela com capacidade de edição do arquivo de configuração. E alteração no menu e na Main Controller para tratar a tela.

Fonte: O próprio Autor.

7.1.3 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Neste tópico são listados os requisitos funcionais e não funcionais levantados para os sprints já desenvolvidos. Seguindo a metodologia incremental, foi realizado o levantamento de requisitos para cada sprint de acordo com as funcionalidades desenvolvidas. O conjunto de todos os requisitos até o atual sprint estão listados abaixo.

7.1.3.1 REQUISITOS FUNCIONAIS

RF001 - O usuário poderá cadastrar Posições do braço robótico;

RF002 - O usuário poderá deletar uma Posição já cadastrada;

RF003 - O usuário poderá editar uma Posição já cadastrada;

RF004 - O usuário poderá listar todas as Posições cadastradas;

RF005 - O usuário poderá cadastrar uma Ação listando quais posições compõem essa Ação e em qual ordem;

RF006 - O usuário poderá deletar uma Ação já cadastrada;

RF007 - O usuário poderá editar uma Ação já cadastrada;

RF009 - O usuário poderá listar todas as Ações cadastradas;

RF010 - O usuário poderá adicionar Posições em uma Ação;

RF011 - O usuário poderá remover Posições de uma Ação;

RF012 - O usuário poderá Alterar o índice de Posições de uma Ação;

RF013 - O usuário poderá, durante o cadastro de Ação, simular o uso da Ação cadastrada;

7.1.3.2 REQUISITOS NÃO FUNCIONAIS

RNF001 - O Software deve rodar nos principais sistemas operacionais do mercado (Windows, Linux e Mac Os X)

RNF002 - As funcionalidades devem ser fácies de acessar

RNF003 - A aplicação deve listar as portas seriais disponíveis de acordo com o sistema operacional

RNF004 - As principais funcionalidades do software como Execução de Testes e Scripts devem estar em um menu na tela principal para fácil acesso;

RNF005 - Ao iniciar o software o sistema deve inicializar as últimas configurações de porta de comunicação e dados do banco de dados;

RNF006 - No cadastro de posições o sistema deve dar feedback da posição cadastrada automaticamente no braço robótico, para facilitar o cadastro de posição;

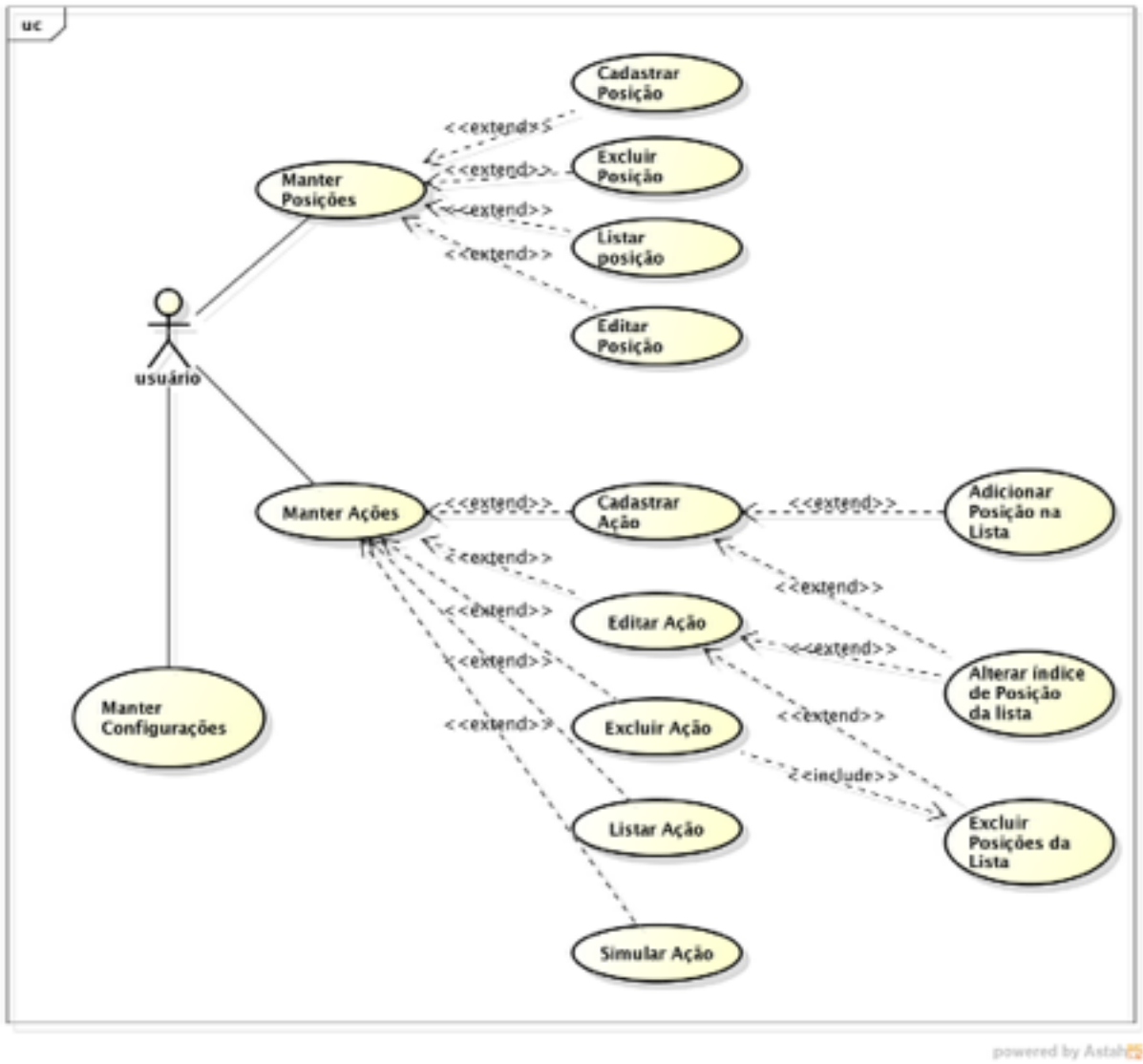
RNF007 - No cadastro de posição o sistema deve permitir que outra posição já cadastrada seja usada como base para uma nova posição;

RNF008 - O software deve prever o uso de diferentes tipos de braço robóticos com até 5 articulações;

7.1.4 DIAGRAMA DE CASOS DE USO

A seguir, na figura 6, são listados os casos de uso deste projeto até o presente momento. Estes casos de uso demonstram as funcionalidades do sistema.

Figura 6 - Diagrama de Casos de Uso

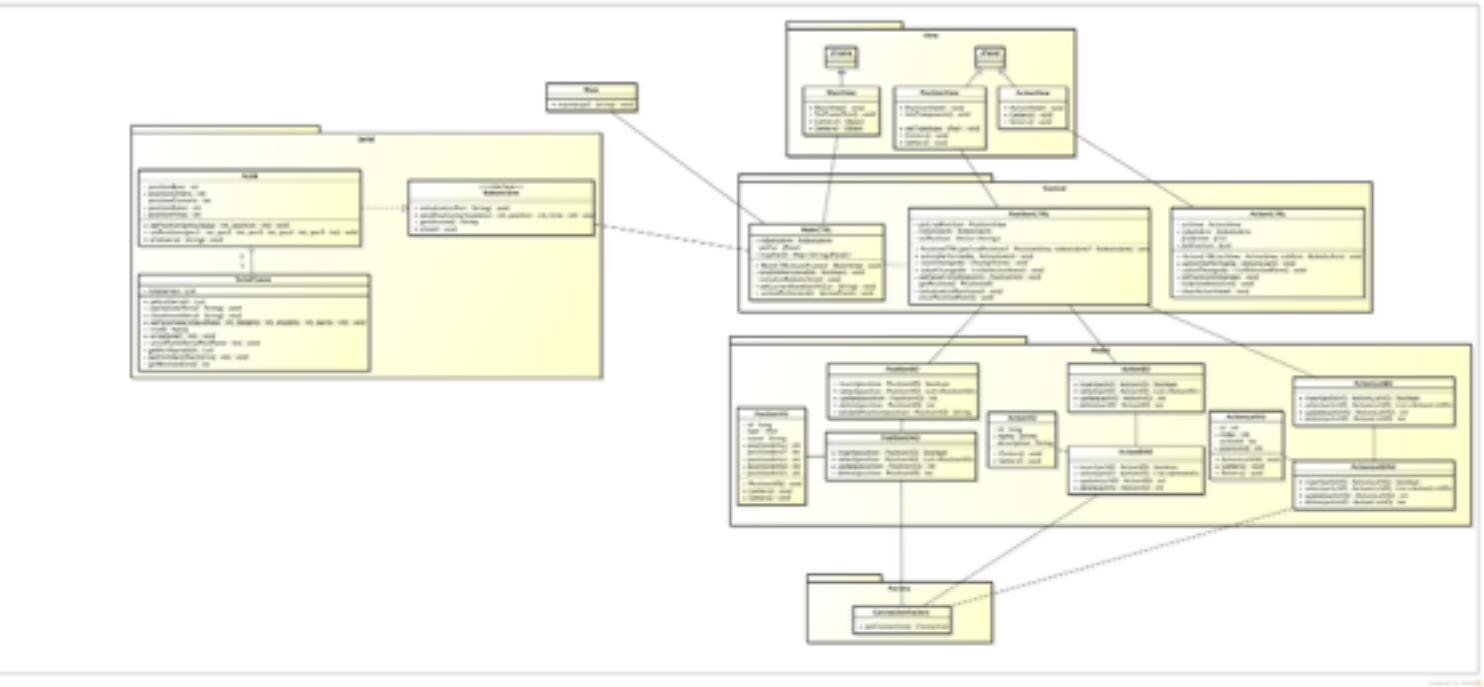


Fonte: O próprio Autor

7.1.5 DIAGRAMA DE CLASSES

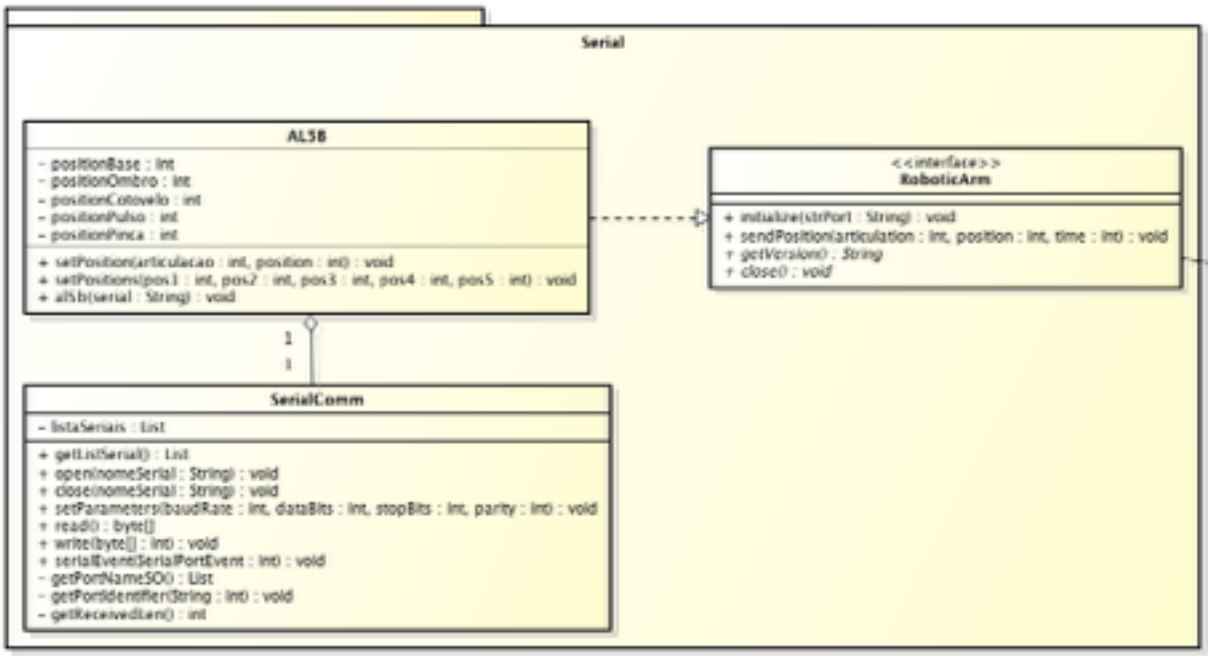
Nesta seção é apresentado o diagrama de classes do sistema de forma macro, que por motivos de formatação teve seus módulos apresentados separadamente nas figuras 8 a 13. A figura 7 abaixo é meramente ilustrativa, afim de demonstrar que o sistema está dividido em camadas. Todas as camadas desta figura serão explicadas a seguir.

Figura 7 - Visão Macro do Diagrama de Classes Atual



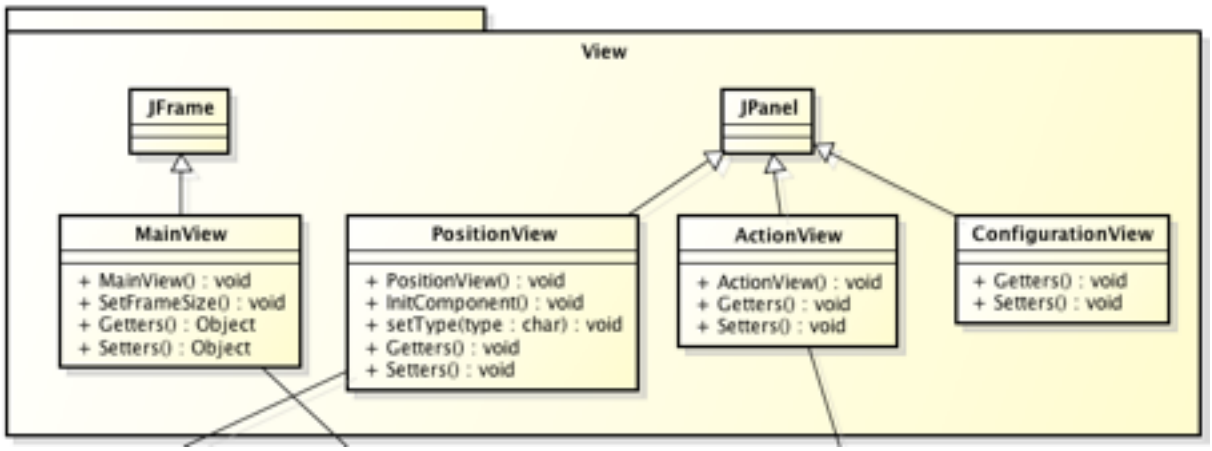
Fonte: O próprio Autor

Figura 8 - Diagrama de Classes: Módulo Serial



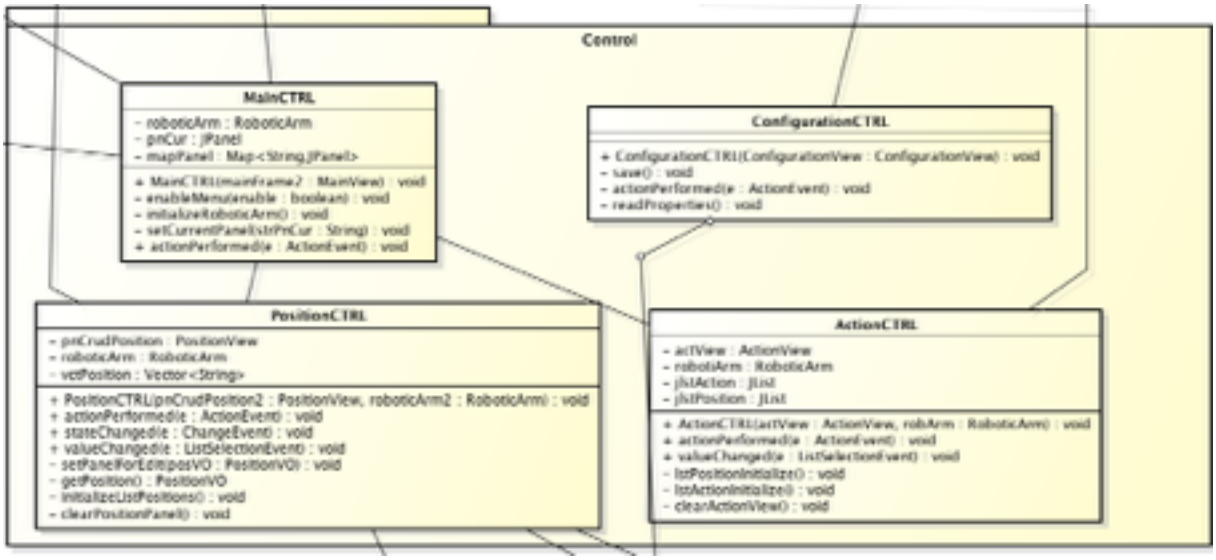
Fonte: O próprio Autor

Figura 9 - Diagrama de Classes: Views



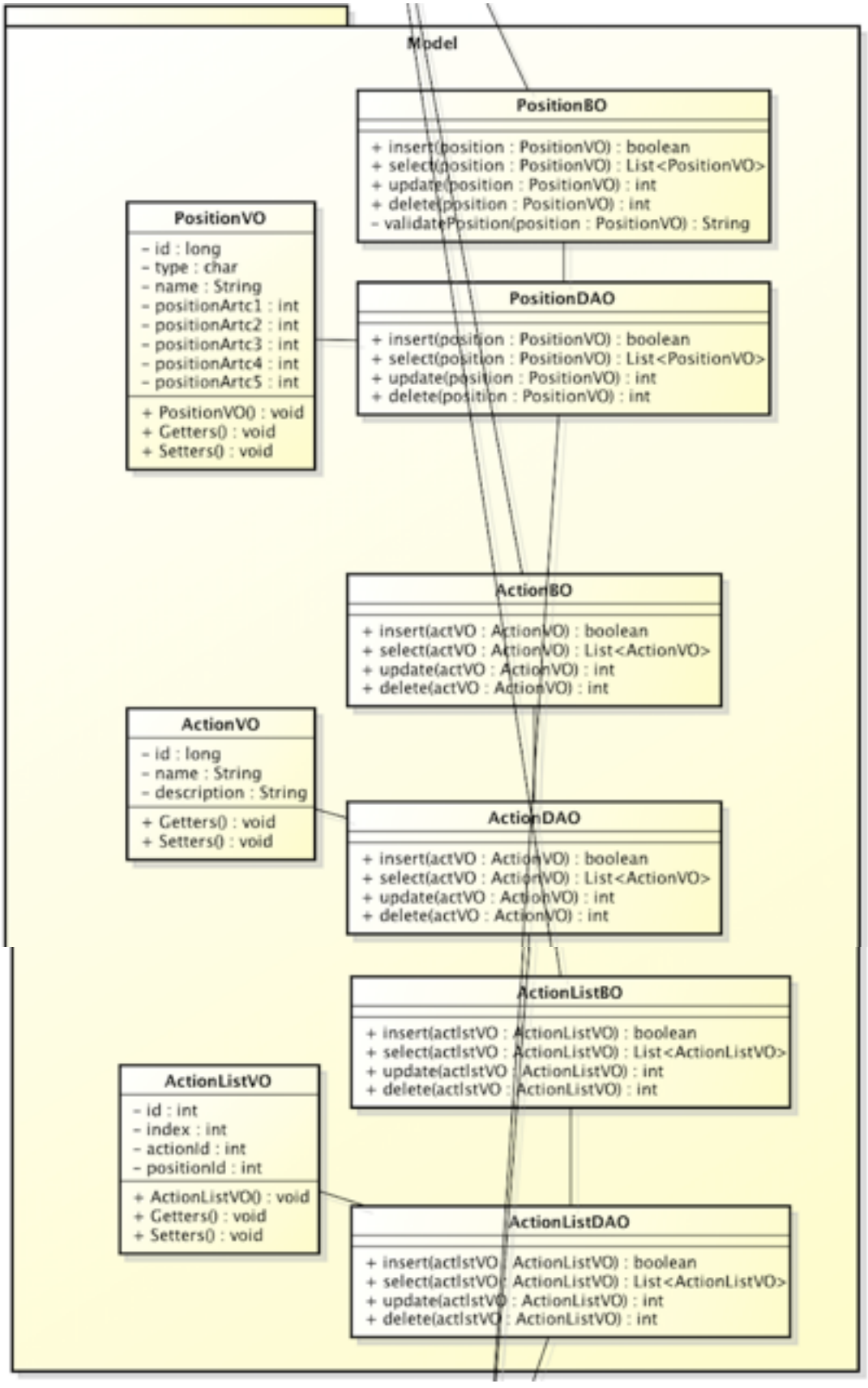
Fonte: O próprio Autor

Figura 10 - Diagrama de Classes: Controllers



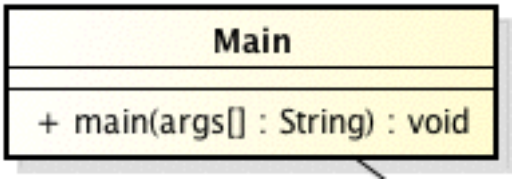
Fonte: O próprio Autor

Figura 11 - Diagrama de Classes: Models



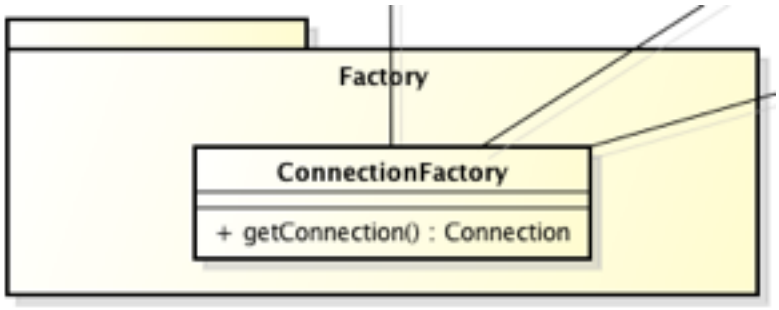
Fonte: O próprio Autor

Figura 12 - Diagrama de Classes: Main



Fonte: O próprio Autor

Figura 13 - Diagrama de Classes: Factory



Fonte: O próprio Autor

7.2 MODELAGEM DE DADOS

Nesta seção serão apresentados o modelo lógico de Banco de Dados e o Arquivo de Configuração do sistema.

7.2.1 MODELO LÓGICO DE BANCO DE DADOS

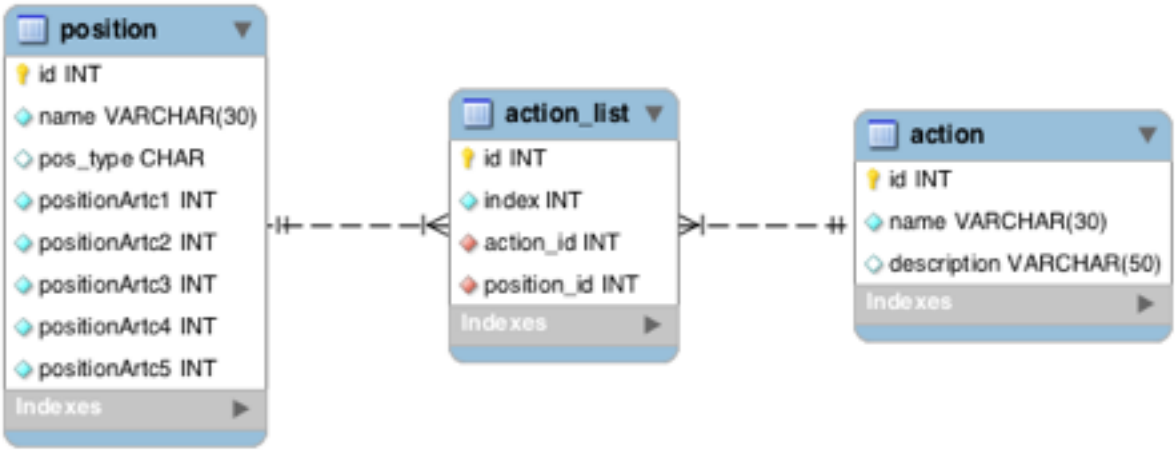
A seguir será apresentado o Modelo Lógico do Banco de Dados no padrão Crows Foot, popularmente conhecido como “Pé de Galinha”. O modelo de banco de dados da Figura 14, apresenta as tabelas atuais do sistema.

A tabela “position" é a entidade que guarda as posições cadastradas para o braço robótico, contém um ID único para cada posição, um nome e a posição para cada articulação do braço robótico.

A tabela "action" guarda as Ações cadastradas. Contém um ID único para referências cada ação, um nome para a ação e sua descrição.

A tabela “action_list” é responsável pela lista de Posições que cada ação possui, ela possui a referência de uma posição existente, de uma ação existente e um índice que indica a ordem na qual a posição foi cadastrada para aquela ação.

Figura 14 - Modelo ER



Fonte: O próprio Autor

7.2.2 ARQUIVO DE CONFIGURAÇÃO

Este arquivo é responsável por guardar as configurações do software. Nele são listados dados como: porta serial de controle do braço robótico, porta serial de debug, dados de acesso a banco de dados.

Em Java utiliza-se arquivos com a extensão “.properties" para descrever as configurações do sistema. A figura 15 apresenta a estrutura do arquivo “config.properties" que é o arquivo de propriedades deste projeto.

Figura 15 - Arquivo properties de configuração

```
prop.serial.interface.name = tty.usbserial
prop.serial.interface.baudrate = 115200
prop.serial.interface.parity = 0
prop.serial.interface.databits = 8
prop.serial.interface.stopbits = 1
prop.serial.debug.name =
prop.serial.debug.baudrate =
prop.serial.debug.parity =
prop.serial.debug.databits =
prop.serial.debug.stopbits =
prop.db.type = postgresql
prop.db.user = armani
prop.db.password = DB_SIHTS
prop.db.ip = localhost
prop.db.port = 5432
prop.db.database = DB_SIHTS
```

8 FUNCIONAMENTO DO SISTEMA

Neste capítulo são apresentadas as funcionalidades já desenvolvidas do sistema, com apresentação das telas do sistema e explicação do seu funcionamento.

8.1 INTERFACE DE TESTE DE SERIAL

A figura 16 - Software de teste da interface serial, apresenta a interface do software desenvolvido durante o Sprint 2 para testar as funcionalidades da Interface Serial.

Através desta interface o usuário, neste caso o desenvolvedor, pode testar se a interface serial trata corretamente a listagem de seriais de acordo com o sistema operacional no qual o software roda, a conexão com a porta serial e o protocolo de comunicação com o Kit AL5B da Lynxmotion.

Em um componente dropdown são listadas as portas seriais presentes no sistema. Ao escolher a porta o usuário pode conectar a mesma pressionando o botão “Conectar”. Após conectado na porta o sistema permite a escolha da articulação a qual o sistema irá interagir. Através de um componente Slider o usuário escolhe uma posição para a articulação escolhida, que é enviada via serial ao pressionar o botão “Mover”.

Figura 16 - Software de teste da interface serial

Fonte: O próprio Autor.

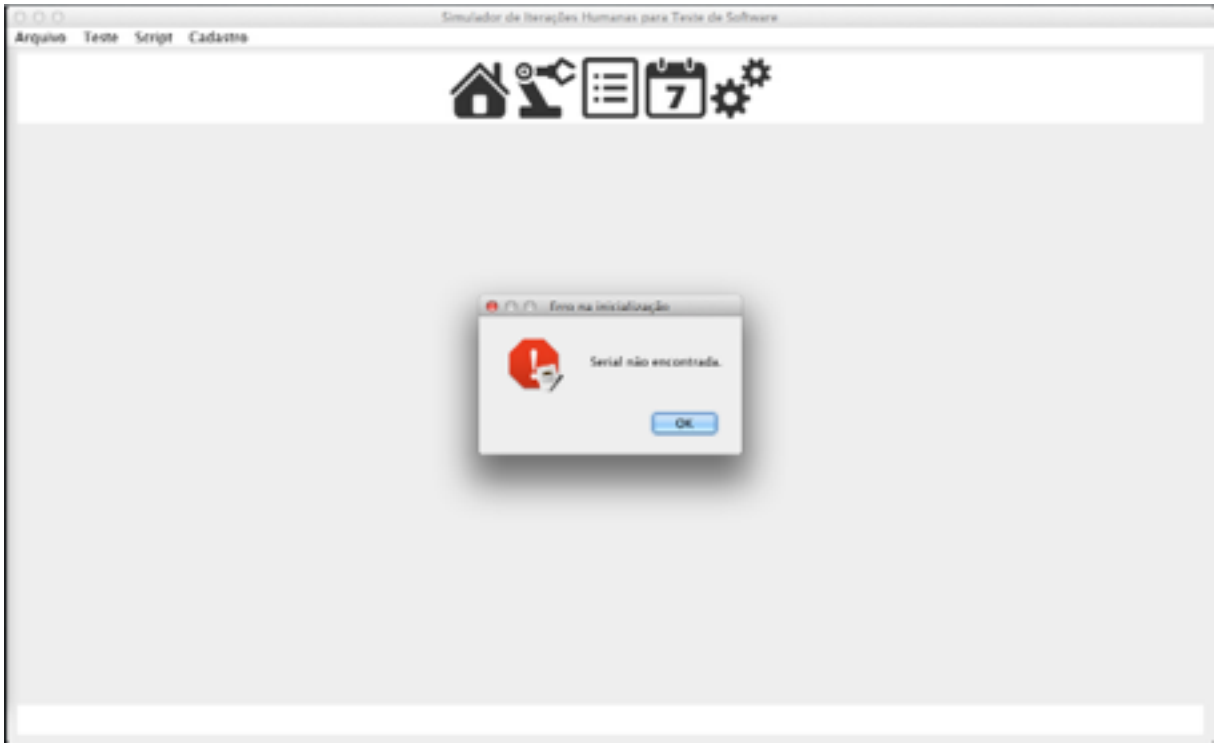
8.2 TELA INICIAL DO SISTEMA

A tela Inicial do sistema apresenta uma barra de Menus com as opções Arquivo (Configuração, Sair), Teste (Novo, Editar, Executar), Script (Novo, Editar, Executar) e Cadastro (Ação, Posição, Objeto). Há também um painel de menu rápido e uma área de notificações/visualização de funcionalidades.

Na área de notificações e visualização de funcionalidades são apresentados as excessões que ocorrerem e também os painéis com as funcionalidades do sistema.

A figura 17 apresenta a tela principal do sistema demonstrando a apresentação de uma mensagem de erro indicando que o sistema, ao inicializar, não encontrou a porta serial configurada.

Figura 17 - Tela Inicial do Sistema Apresentando erro na porta serial



Fonte: O próprio Autor.

8.3 CONFIGURAÇÕES DO SITEMA

A tela de configurações do sistema, apresentada na figura 18, reflete o arquivo de configuração “config.properties" e permite a alteração das configurações da serial de controle do braço robótico, da serial de debug e dos dados de conexão com o banco de dados. Ao selecionar a tela de configuração automaticamente são listados todos os dados do arquivo de configuração.

Ao pressionar “SALVAR" os dados são atualizados no arquivo e o sistema reinicializa as interfaces de comunicação e de conexão com o banco de dados.

Figura 18 - Tela de configuração do sistema

Fonte: O próprio Autor.

8.4 CADASTRO DE POSIÇÕES

O objetivo do cadastro de Posições é permitir ao usuário do sistema definir posições para as 5 articulações do braço robótico. Estas posições serão utilizadas posteriormente para, em grupo de posições definir uma Ação.

Ao acessar a tela de Cadastro de Posição o usuário tem uma lista de posições já cadastradas a sua esquerda e a direita o formulário de cadastro de posição. Para cadastrar uma nova posição basta preencher o campo "Posição" com o nome da posição que está cadastrando, alterar a posição dos sliders referentes a posição das articulações e pressionar “Salvar”.

Sempre que o usuário alterar a posição de um slider, ao soltar o componente, o sistema envia a alteração para o Braço Robótico mostrando assim a posição real que se está cadastrando.

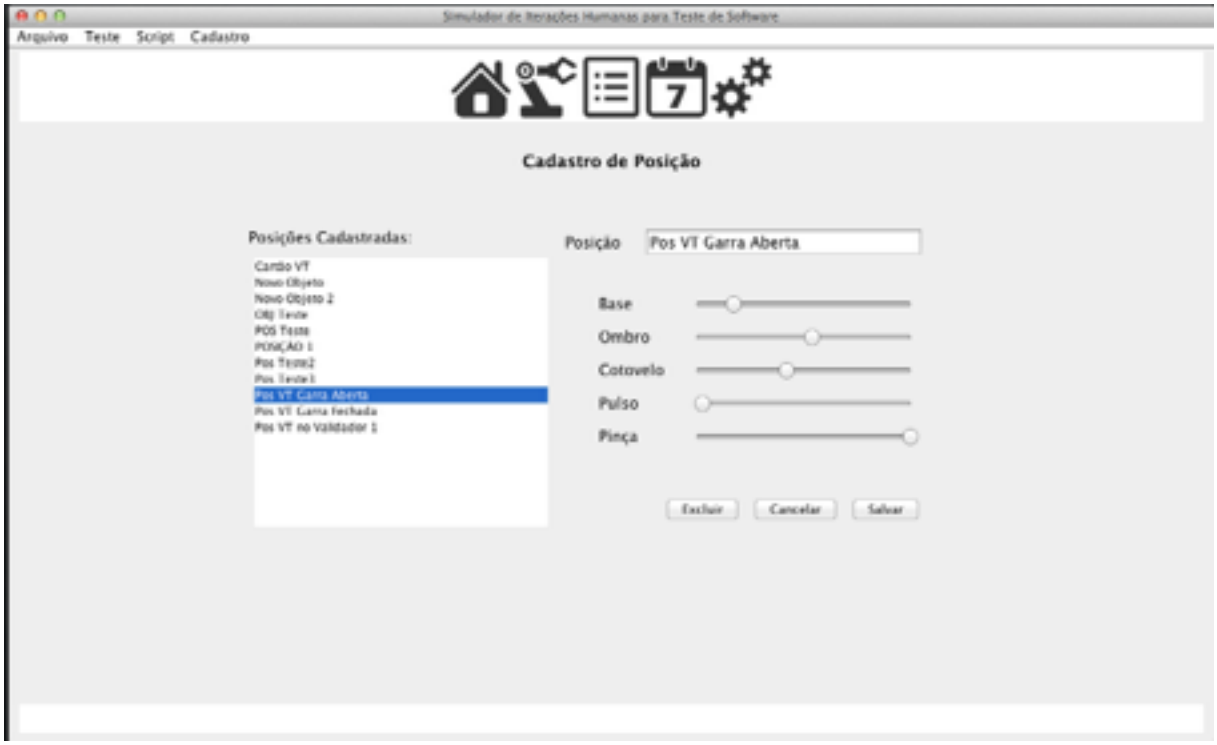
Ao pressionar salvar o sistema guarda a nova posição no banco de dados e automaticamente lista a nova posição no quadro “Posições Cadastradas”.

Se o usuário clicar em uma posição já cadastrada, os dados da mesma são expostos no formulário de cadastro e é habilitado o botão “Excluir”, este botão da a possibilidade de exclusão da posição selecionada do banco de dados do sistema.

Ao ser pressionado o botão “Cancelar" o sistema irá limpar o formulário de cadastro e retirar a seleção da posição corrente.

A figura 19 apresenta a tela de Cadastro de Posição no modo de edição de uma posição.

Figura 19 - Tela de cadastro de posição



Fonte: O próprio Autor.

8.5 CADASTRO DE AÇÕES

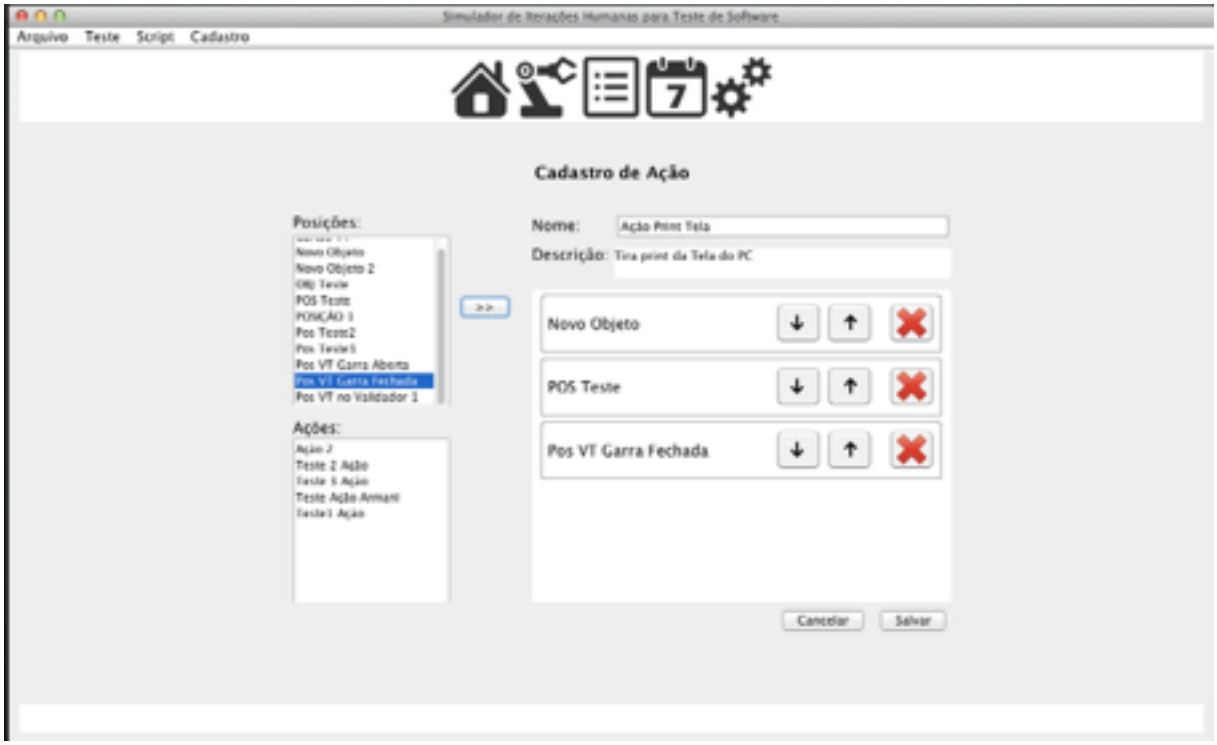
A tela de cadastro de Ação, figura 20, serve para definir Ações realizadas pelo braço robótico. Uma ação, para este sistema, consiste na união de uma sequência finita de posições. Logo, para cadastrar uma Ação, o usuário cadastra um nome, uma descrição e uma lista de posições ordenada que, quando executadas em ordem, geram uma ação real.

A figura 19 apresenta a tela de Cadastro de Ação, que é composta por um campo de nome, um campo de descrição e um painel em branco onde serão listadas as posições da ação corrente. A esquerda encontram-se duas listas a superior contendo todas as Posições já cadastradas e a inferior contendo todas as Ações já cadastradas.

Durante o cadastro de uma ação, para adicionar uma posição à ação que se está cadastrando, deve-se selecionar a posição na lista de posições e pressionar o botão “>>”. Após pressionado o botão a posição aparece automaticamente no painel de posições da ação, logo abaixo do campo descrição.

Para editar ou excluir uma Ação já cadastrada basta selecioná-la na lista de Ações e posteriormente pressionar o botão relacionado ao evento que se deseja executar.

Figura 20 - Tela de Cadastro de Ação



Fonte: O próprio Autor.

9 VALIDAÇÃO

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

9.1 ESTRATÉGIA

A validação do Simulador de Iterações Humanas para Testes de Software ocorre 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. Este tipo de validação ocorrerá após o desenvolvimento do mínimo produto executável, no caso quando estiver pronto o cadastro de Testes e o motor de execução destes.

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.

9.2 CONSOLIDAÇÃO DE DADOS

Neste item serão apresentados os dados de validação do software. Até a presente data há apenas a validação dos sprints desenvolvidos, 1 ao 5, que segue abaixo no Quadro 4 - validação dos sprints. As demais validações ocorrerão na implantação do projeto em ambiente de testes da ATP, o que ocorrerá nos próximos sprints.

Após a implantação do software na ATP 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.

Quadro 3 - Validação dos Sprints

Sprint	Tipo Validação	Observação	Resultado
Sprint 1	Avaliação do plano de trabalho pelo Senac	Plano de trabalho aceito sem revisões	OK
	Test Comunicação Serial do módulo desenvolvido, envio e recepção com equipamento externo	Necessários alguns ajustes iniciais, mas após funcionou corretamente.	OK
	Teste da interface de braço robótico e comunicação com Kit AL5b utilizando software de desenvolvido para teste	Comunicação funcionando e interface desenvolvida se mostrou eficaz.	OK
Sprint 2	Teste do sistema ser multiplataforma. (Windows, Linux e Mac Os X)	Verificada a diferenciação da interface gráfica do software desenvolvido para teste seguindo padrões das plataformas. Necessários cuidados futuros.	OK
Sprint 3	Teste de cadastro, edição e exclusão de Posição e tentativa de simulação de problemas.	Encontrada necessidade de enviar dados ao braço serial somente quando componente slider for solto. Corrigido!	OK
	Execusão nas plataformas (Windows, Linux e Mac Os X)	Sem problemas	OK
	Verificação manual no banco para confirmar inclusão de dados.	Sem problemas	OK
Sprint 4	Simulação de inicialização do software sem serial	Apresentou mensagem corretamente.	OK
	Refeitos os testes do sprint 3 por causa de refatoração do código em função de alteração para padrão MVC	Pequenas correções na estrutura MVC corrigidas.	OK
	Test de funcionamento de Menus da tela principal	Sem problemas	OK
	Teste de redimensionamento da tela principal, como e sem painéis de funções.	Erro ao redimensionar a tela, não está reposicionando os painéis de função. Aguardando refatoração de código com correção.	-
	Teste de cadastro de ação com verificação de inclusão no banco de dados.	Sem problemas	OK
Sprint 5	Teste de edição e exclusão de Ação	Ao excluir Ação não estava excluindo lista de ação também. Problema Corrigido!	OK

Sprint	Tipo Validação	Observação	Resultado
	Teste de alteração de configuração	Apresentou necessidade de reinicializarão de algumas funcionalidades. Corrigido!	OK

As funcionalidades desenvolvidas até o momento, segundo os testes de validação de sprint, estão funcionais e atendem as expectativas. A única excessão encontrada e ainda não corrigida está no redimensionamento da tela principal, que não está reposicionando o painel central corretamente, aquele que contém as funcionalidades. Este problema será corrigido no próximo sprint em uma ação de refatoração do código da tela principal.

Segundo o Product Backlog, apresentado no item 7 deste relatório, foram desenvolvidas até o momento 7,5 funcionalidade de 17 previstas. Ou seja, em torno de 44% do projeto está concluído.

Estes números mostram que o ritmo de desenvolvimento está de acordo com o esperado, que era em torno de 50% do sistema pronto até dia 02/06/2014. Tendo em vista que o período de entrega dos 56% restantes é maior que o decorrido até o momento, espera-se alcançar a conclusão total do projeto e também sua validação dentro do prazo estipulado no cronograma, novembro de 2014.

10 CRONOGRAMA

A Seguir é apresentado o cronograma do projeto, com as sprints já realizadas e a previsão de período de execução das demais sprints, sem conteúdo programado. O cronograma para o segundo semestre de desenvolvimento se apresenta maior e capaz de suportar o fluxo de trabalho ligeiramente maior que no primeiro período de desenvolvimento.

Quadro 4 - Cronograma

Texto

Atividade	Produto	Data Prevista	Data Realizada	Descrição
	Aceite TCC1	17/03/2014	17/03/2014	Entrega do formulário de aceite
Sprint 1	Plano de Trabalho	31/03/2014	31/03/2014	Desenvolvimento e entrega do Plano de Trabalho
Sprint 2	Interface serial com kit AL5B	14/04/2014	16/04/2014	Desenvolvido uma interface de controle de serial em java. Através desta interface foi desenvolvido o controle do kit AL5B.
Sprint 3	Estruturação MVC	28/04/2014	30/04/2014	Estruturação do projeto no padrão MVC e criação da base para o cadastro de posições
Sprint 4	Cadastro de Posições	12/05/2014	16/05/2014	Implementação da tela de cadastro de posições com gravação em banco de dados
Sprint 5	Cadastro de Ações	26/05/2014	29/05/2014	Interface de cadastro de ações e interface de configurações.
	Relatório Parcial	02/06/2014	30/06/2014	Finalização do desenvolvimento do Relatório Parcial. **Entrega do relatório parcial**
	Banca TCC1	09/06/2014		Desenvolvimento da Apresentação do Projeto. **Banca Final TCC1.**
Sprint 6		05/08/2014		
Sprint 7		19/08/2014		
	Aceite TCC2	29/08/2014		Entrega do fomrmulário de aceite
Sprint 8	-	12/09/2014		
	Relatório de Projeto	22/09/2014		Entrega do Relatório de projeto atualizado
Sprint 9	-	26/09/2014		
	Seminário de Andamento	29/09/2014 a 03/10/2014		Seminário de andamento do projeto
Sprint 10	-	10/10/2014		
Sprint 11	-	24/10/2014		

Atividade	Produto	Data Prevista	Data Realizada	Descrição
Sprint 12	-	07/11/2014		
	Relatório Final	17/11/2014		Entrega do relatório final do projeto
Sprint 13	-	21/11/2014		
	Banca Final	24/11/2014 a 28/11/2014		Banca Final
	Relatório	08/12/2014		Entrega da versão final do relatório

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.