Instituto Federal de Educação, Ciência e Tecnologia Fluminense

Programa de Pós-graduação em Sistemas Aplicados à Engenharia e Gestão

PROJETO DE DISSERTAÇÃO

ORQUESTRAÇÃO E AUTOMAÇÃO DA REDE INTEGRADA DE RASTREAMENTO DE SATÉLITES (RIBRAS)

LUCAS RODRIGUES AMADURO

2018

Instituto Federação de Educação, Ciência e Tecnologia Fluminense
Programa de Pós-graduação em Sistemas Aplicados à Engenharia e Gestão

ORQUESTRAÇÃO E AUTOMAÇÃO DA REDE INTEGRADA DE RASTREAMENTO DE SATÉLITES (RIBRAS)

LUCAS RODRIGUES AMADURO

Prof. D. Sc. Luiz Gustavo Lourenço Moura
(Orientador)

Prof. D. Sc. Rogerio Atem de Carvalho
(Co-Orientador)

Projeto submetido como requisito para elaboração da dissertação para obtenção do grau de **Mestre** no Programa de Pós-graduação em Sistemas Aplicados à Engenharia e Gestão, Área de Concentração em Sistemas Computacionais.

Campos dos Goytacazes,RJ
Maio de 2017

Instituto Federação de Educação, Ciência e Tecnologia Fluminense
Programa de Pós-graduação em Sistemas Aplicados à Engenharia e Gestão

LUCAS RODRIGUES AMADURO

Projeto submetido como requisito para elaboração da dissertação para obtenção do grau de **Mestre** no Programa de Pós-graduação em Sistemas Aplicados à Engenharia e Gestão, Área de Concentração em Sistemas Computacionais.

PROJETO APRESENTADO EM 02/03/2018

BANCA EXAMINADORA

Prof. Luiz Gustavo Lourenço Moura, D. Sc. (Orientador)
Instituto Federal Fluminense – IFF

Prof. Rogerio Atem de Carvalho, D. Sc. (Co-Orientador)
Instituto Federal Fluminense – IFF

Prof. Breno Fabrício Terra Azevedo, D. Sc.
Instituto Federal Fluminense – IFF

Prof. William da Silva Vianna, D. Sc.
Instituto Federal Fluminense – IFF

RESUMO

Introdução: Atualmente há uma tendência para desenvolver constelações de satélites pequenos, em vez de um único grande satélite (CARVALHO, R. A. et al., 2014). Para atender a demanda de coleta de dados a partir de uma constelação de satélites, o CRSEA (Centro de Referência para Sistemas Embarcados e Aeroespaciais) do Instituto Federal Fluminense (IFF, Brasil) está desenvolvendo a RIBRAS (Rede Integrada Brasileira de Estações de Rastreamento de Satélites), com financiamento da Agência Espacial Brasileira (AEB) e apoio do Ministério da Educação (MEC). **Objetivo:** Este trabalho tem como objetivo o desenvolvimento de um software que auxilie na automação da distribuição de trabalhos e de rastreamentos entre as estações terrestres que estão distribuídas pela rede da RIBRAS. **Revisão Bibliográfica:** As Estações Terrestres trabalharão de forma sincronizada seguindo um plano de trabalho gerado anteriormente pelo software desenvolvido. Para atingir esses objetivos, cada estação terrestre é equipada com duas torres independentes, uma para a banda S e outra para as comunicações UHF e VHF. **Metodologia:** A metodologia é composta pelas etapas: pesquisa bibliográfica, Iniciação da implementação e elaboração do módulo Distributor, construção e desenvolvimento do módulo Distributor no lado do servidor, testes de desenvolvimento, validação do software desenvolvido e proposto. **Resultados:** Com a o início da rede de estações terrestres autônomas, o rastreamento de satélites seria potencializado drasticamente, com a perda de passagens caindo para quase zero, além de suprir a necessidade de uma mão de obra qualificada para o rastreio de satélites. **Conclusão:** Contribuir de forma significativa com o rastreamento de satélites, implementando um módulo de software de automação em uma rede de estações terrestres, como a RIBRAS.

Palavras-chave: RIBRAS, Satélites, Estações Terrestres, Distributor.

ABSTRACT

Introduction: Currently there is a tendency to develop constellations of small satellites instead of a single large satellite (CARVALHO, R. A. et al., 2014). In order to meet the demand for data collection from a constellation of satellites, the CRSEA (Center of Reference for Embedded and Aerospace Systems) of the Federal Fluminense Institute (IFF, Brazil) is developing RIBRAS (Brazilian Integrated Network of Satellites), with funding from the Brazilian Space Agency (AEB) and support from the Ministry of Education (MEC). **Objective:** The objective of this work is the development of software that assists in automation of the distribution of jobs and of tracing between the ground stations that are distributed by the RIBRAS network. **Bibliographic Review:** Ground Stations will work in a synchronized way following a work plan previously generated by the developed software. To achieve these objectives, each ground station is equipped with two independent towers, one for the S-band and the other for UHF and VHF communications. **Methodology:** The methodology is composed by the steps: bibliographic research, initiation of the implementation and elaboration of the Distributor module, construction and development of the Distributor module on server side, development tests, analysis and data collection between station and satellites, validation of the software developed and proposed. **Results:** With the network of autonomous ground stations, satellite tracking would be dramatically boosted, with the loss of passages dropping to almost zero, in addition to meeting the need for a skilled labor force to track satellites. **Conclusion:** Contribute in a significant way to the tracking of satellites, implementing an automation module software in a network of ground stations, such as RIBRAS.

Keywords: RIBRAS, Satellite, Ground Station, Distributor.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE SIGLAS

RIBRAS	Rede Integrada Brasileira de estações de Rastreamento de Satélites
AEB	Agência Espacial Brasileira
MEC	Ministério da Educação
CRSEA	Centro de Referência para Sistemas Embarcados e Aeroespaciais
PICG	Polo de Inovação de Campos dos Goytacazes
UHF	Ultra High Frequency
VHF	Very High Frequency
GENSO	Global Educational Network for Satellite Operations
NORAD	North American Aerospace Defense Command
SGP4/SDP4	Simplified perturbations models
TLE	Two-line element set
ISEB	Instituto Superior de Estudos Brasileiros
COSMIAC	Space Electronics Center in University of New Mexico
NOAA	National Oceanic and Atmospheric Administration
GPL	General Public License
TDD	Test Driven Development
LEO	Low Earth Orbit
AUS	Authentication Server
GSS	Ground Station Server
MCC	Misson Client Control
DSN	Deep Space Network
SOCC	Spacecraft Operation Control Center
TT&C	Telemetry, Tracking and Control

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Na última década, o voo espacial educacional e acadêmico tornou-se mais e mais popular. A introdução do padrão CubeSat (WAYDO; HENRY; CAMPBEL, 2002) fez o design, construção e lançamento de satélites acessíveis para instituições acadêmicas de todos os tamanhos. Um dos requisitos para o lançamento de um satélite próprio é uma estação terrestre local de rastreio de satélites (TULI; ORR; ZEE, 2006). A maioria das missões espaciais não-comerciais utilizam as faixas de frequências de rádio amador sem fins lucrativos (KARN; PRICE; DIERSING, 1985) para a comunicação nos dois sentidos, devido ao fato de que as estações terrestres dos rádio-amadores (BAKER; JANSSON, 1994) também são capazes de se comunicar com veículos espaciais para envio e recebimento de dados nestas frequências específicas.

Como nano-satélites e satélites de pequeno porte, como CubeSats, operam com orçamentos de energia muito mais baixos (menos de um Watt) e seu hardware não se destina a resistir radiações muito elevadas como a presente no cinturão de Van Allen (HESS, 1968), estes operam exclusivamente na Baixa Órbita Terrestre (LEO) (CAKAJ; KEIM; MALARIC, 2007). Como consequência, o período médio de comunicação entre a estação terrestre e de veículos espaciais é de meia hora por dia.

Para combater esses fatores limitantes, redes de estações terrestres são desenvolvidas para que se estenda o prazo máximo possível de comunicação à poucas horas por dia. O projeto mais popular é a Rede de Educação Global para Operações de Satélite (GENSO) (PREINDL; PAGE; NIKOLAIDIS, 2008). Além do objetivo principal da rede, o aumento da quantidade total de dados que passam entre estações terrestres e satélites por fornecimento e orquestração de hardware da estação terrestre e comunicação através da Internet, as estações participantes também coletam dados de potência do sinal durante passagens de satélite (PREINDL et al., 2008).

Atualmente, há uma tendência a desenvolver constelações de satélites pequenos em vez de um único grande satélite (CARVALHO, R. A. et al., 2014). Para cumprir com a demanda de comunicação de uma constelação de satélites, o CRSEA (Centro de Referência em Sistemas Embarcados e Aeroespaciais do Instituto Federal Fluminense (IFF, Brasil) está

desenvolvendo a RIBRAS (Rede Integrada Brasileira de Estações de Rastreamento de Satélites), com financiamento da Agência Espacial brasileira (AEB) e o apoio do Ministério da Educação (MEC).

A RIBRAS consiste de um conjunto de estações terrestres para a localização de satélites distribuídos pelo território brasileiro. Este projeto é composto pela concepção e integração das estações e suas torres e o desenvolvimento dos softwares de automação, controle e orquestração. O sistema em si é um conjunto de softwares que visam controlar a rede de estações terrestres que receberá dados dos satélites que sobrevoam o território nacional.

1.2 Objetivo Geral e Específicos

O objetivo geral:

- O desenvolvimento do Distributor, um módulo do software que permite que a RIBRAS tenha uma maior automação, distribuição e orquestração de seus rastreamentos de satélites.

Os objetivos específicos:

- Diminuição da necessidade de uma mão de obra qualificada para a realização de um rastreamento programado;
- Melhora significativa na qualidade do rastreamento de satélites e constelações de satélites.

1.3 Justificativa

Um problema em comum de todas as estações de rastreamento de satélites é a necessidade de uma mão de obra qualificada de um operador para todas as vezes que são necessários os rastreios de um satélite ou de uma constelação de satélites tendo uma perda significativa de passagens e na transmissão dos dados para as estações terrestres. Com a rede de estações terrestres autônomas, o rastreamento de satélites seria potencializado

drasticamente, com a perda de passagens caindo para quase zero, além de suprir a necessidade de uma mão de obra qualificada para o rastreio de satélites.

As estações terrestres irão funcionar de forma sincronizada, na sequência de um plano de trabalho gerado anteriormente, para se certificar de que a quantidade máxima de dados serão coletados durante os sobrevoos. O software é organizado para o modelo de comunicação mestre / escravo, com cada um dos servidores (master) e os pacotes de clientes (escravos) com três módulos: Scheduler, Distribuidor e um Controller, para o servidor, e Controlador de tarefas (Job Controller), Controlador de antenas (Antenna Controller) , e Controlador de comunicação (Communication Controller) para cada cliente - as estações terrestres. As Estações terrestres adquirem os dados dos experimentos embarcados do satélite e dos seus dados de telemetria, e depois transfere esses dados para os usuários. O foco deste trabalho é o desenvolvimento do módulo Distributor do software.

1.4 Metodologia

Para atingir os objetivos do projeto, inicialmente foi realizada uma revisão da literatura para identificar quais trabalhos relacionados foram realizados na academia. Foi feito também um levantamento bibliográfico afim de encontrar algum trabalho com o mesmo intuito.

A partir dessas tarefas realizadas, a metodologia será desenvolvida nas seguintes etapas:

- Pesquisa bibliográfica sobre Sistemas Distribuídos em estações terrestres;
- Iniciação da implementação e elaboração do módulo de software Distributor;
- Construção e desenvolvimento do módulo de software de auxílio de automação das estações terrestres;
- Testes de desenvolvimento do software utilizando TDD e objetos Mock;
- Validação do software desenvolvido e proposto.

2 REVISÃO BIBLIOGRÁFICA

2.1 Técnicas de Desenvolvimento

Em desenvolvimento de software, é necessário, para uma boa organização e trabalho do código escrito no desenvolvimento do software, metodologias e técnicas de desenvolvimento que combinem e se encaixem melhor no seu plano de trabalho e para a finalidade do software proposto. Em seguida, serão explicadas as técnicas usadas para o desenvolvimento e implementação do software proposto deste trabalho.

2.1.1 Test Driven Development (TDD)

Test Driven Development significa escrever os testes antes do código produtivo – em ciclos curtos e contínuos – reforçando o design, o desacoplamento e a eficácia em resolver os problemas que motivaram sua escrita. É uma técnica de desenvolvimento de software que baseia em um ciclo curto de repetições: Primeiramente o desenvolvedor escreve um caso de teste automatizado que define uma melhoria desejada ou uma nova funcionalidade. Então, é produzido código que possa ser validado pelo teste para posteriormente o código ser refatorado para um código sob padrões aceitáveis (BECK, 2003).

Simplesmente, o desenvolvimento orientado por testes destina-se a eliminar o medo no desenvolvimento de aplicações. Enquanto algum medo é saudável acredita-se que os subprodutos do medo incluem programadores tentativos, mal-humorados e pouco comunicativos que são incapazes de absorver críticas criativas. Quando as equipes de programação compram o TDD, eles imediatamente vêem resultados positivos. Eles eliminam o medo envolvido em seus empregos e estão melhor equipados para enfrentar os difíceis desafios que enfrentam. TDD elimina traços tentativos, ensina os programadores a se comunicar, e encoraja os membros da equipe a procurar críticas. Em suma, a premissa por trás do TDD é que o código deve ser continuamente testado e refatorado (BECK, 2003).

2.1.2 Objetos Dublê/Mock

Os objetos dublê/mock, em desenvolvimento de software, são objetos que simulam o comportamento ou as atividades de outros objetos de maneira controlada. Esses objetos

"falsos" são criados para testar como o objeto real irá reagir ou para simular uma resposta esperada do mesmo.

Porém há uma diferença entre os dois. Enquanto um dublê apenas provê respostas prontas para as chamadas que serão feitas durante o teste, o dublê vai mais além e, além de prover as respostas, também valida as chamadas - ele conhece o comportamento esperado do sistema e testa este comportamento.

Em testes, pode-se simular o comportamento de objetos reais complexos e são, portanto, muito úteis quando os objetos reais são difíceis de criar ou impossíveis de serem incorporados no teste de unidade. Por exemplo, em um teste para um objeto que faz uma chamada remota é necessário que do outro lado exista um previsão para a chamada. Neste caso, em vez de usar o objeto real que faz a chamada, usa-se um objeto mock que simula o comportamento do objeto real (FREEMAN; PRYCE, 2009).

2.3 Pequenos Satélites

O termo pequenos satélites começou a ser usado para dar significado aos esforços de miniaturização em cursos de engenharia de satélite. No entanto, o termo pequeno satélite não é restrito a essa definição, dependendo do contexto, é utilizado para uma vasta gama de veículos espaciais. Nos últimos anos, o termo pequeno satélite passou a ser adotado para uma nave espacial com menos de cem quilogramas. Em universidades os termos pico, nano e micro-satélites foram estabelecidos para descrever diferentes tipos de pequenos satélites. Muitas vezes a seguinte classificação é utilizada (HAEUSLER; WIEDEMANN, 2008): “A massa de um pico-satélite é menos de um quilograma, nano- satélites são veículos espaciais entre um e dez quilogramas, enquanto micro satélites têm um peso máximo de cem quilogramas.” Esta classificação é bastante utilizada, no entanto, existem outras definições (BRIEß; SCHILLING, 2008).

Segundo Schmidt (2011):

“Um grande número de pequenos satélites foi lançado com sucesso, os primeiros pico-satélites foram colocados em órbita em 2000, nano-satélites foram lançados muito antes (por

exemplo, OSCAR-1 em 1961), mas nos últimos dez anos, o número aumentou de forma constante devido aos avanços na miniaturização."

Um dos marcos no desenvolvimento de pequenos satélites é a criação do padrão CubeSat, que deu início a um enorme número de projetos em todo o mundo. O conceito de um CubeSat, um pico-satélite com forma cúbica, comprimento lateral de dez centímetros e um peso máximo de um quilograma foi introduzido pela primeira vez pelo professor Jordi Puig-Suari da Califórnia Polytechnique State University (CalPoly) e Robert Twiggs da Universidade de Stanford, o que corresponde a especificação e design do CubeSat foi publicado em 1999 (HEIDT et al., 2000).

Ainda segundo Schmidt (2011):

> "Um primeiro lote de cinco CubeSats foi lançado com um foguete Rokot russo em 2003. Nos últimos sete anos, mais de trinta CubeSats foram colocados em órbita, negligenciando muitos outros projetos atualmente em desenvolvimento ou à espera de lançamento (principalmente a nível universitário)."

Um dos pontos fundamentais para o sucesso do padrão CubeSat é a estrutura padronizada, que é uma interface amplamente adotada e usada em satélites pequenos em desenvolvimento. Devido a essa característica, os fornecedores de lançamento oferecem lugares (slots) de lançamento para CubeSats, vários dispositivos e adaptadores de lançamento para lançamentos sobrepostos já existem, e as oportunidades de lançamento são concedidas a partir das agências espaciais para as universidades.

Segundo Roysam et al. (2014):

> "Graças a sua acessibilidade, instituições acadêmicas, especialmente departamentos de ECE, agora tem acesso sem precedentes ao espaço para educação e pesquisa científica. CubeSats requerem uma variedade de sensores, atuadores e

sistemas computacionais embarcados. Construir um projeto de CubeSat com sucesso junta uma variedade de sub-disciplinas ECE incluindo virtualização de imagens e óticas, computação embarcada, navegação, comunicação e métodos de coleta de energia.”

2.4 Estações Terrestres

Estações terrestres adquirem dados de missões de um veículo espacial e seus instrumentos e transferem-no para os usuários dos dados (WERTZ; LARSON, 2005).

O hardware básico de uma estação terrestre é composto por uma antena para captação e envio de dados, um rádio para realização da comunicação e um conjunto de rotores que posicionam a antena na melhor posição possível para a comunicação. No caso das estações terrestres que estão sendo desenvolvidas para a RIBRAS, elas serão compostas por dois conjuntos de rádio, rotores e antenas, um para Banda-S e outro para VHF/UHF. Conforme pode ser visto na figura 1.

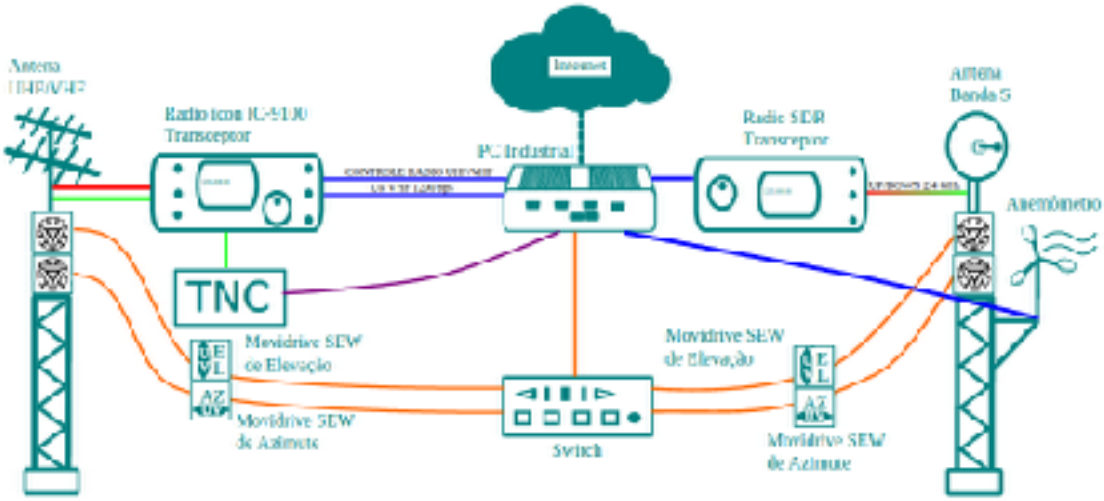


Figura 1 – Estrutura de uma Estação Terrestre da RIBRAS.

Fonte: CRSEA

Nos próximos tópicos serão explicados e detalhados os modelos e softwares usados e implantados em uma estação terrestre.

2.5 Modelos de Estações Terrestres

Dentro da revisão de literatura realizada foram encontrados dois modelos de estações terrestres, um *Standalone*, que lida com a estação de forma individual e outro em rede, que opera mais de uma estação de forma conjunta obedecendo a uma central de controle.

2.5.1 Orbitron

Orbitron é um sistema de rastreamento de satélite para rádio amador e fins observacionais. É também usado por profissionais de meteorologia, usuários de comunicações

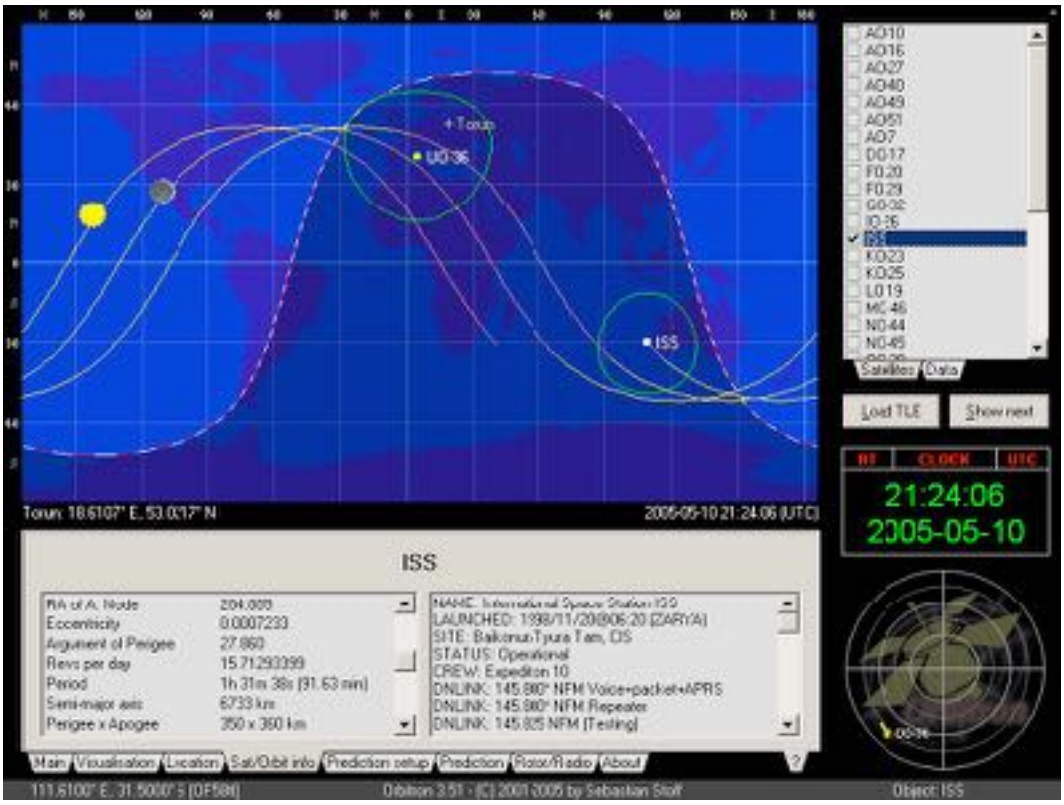


Figura 2 – Interface do Orbitron.

Fonte: (STOFF, 2005)

por satélite, astrônomos e amadores. O aplicativo mostra as posições dos satélites, em determinado momento (em tempo real ou simulada). A figura 2 mostra a interface do orbitron.

As principais funcionalidades do orbitron são modelos de previsão NORAD SGP4 / SDP4, vinte mil satélites podem ser carregados a partir de uma TLE, todos eles podem ser rastreados simultaneamente, modos de apresentação, Modo tempo real / Simulação (controle de tempo livre), Informação de órbitas, radar, interface fácil e flexível, comandos de rádio e rotor.

2.5.2 PREDICT

PREDICT é um programa de rastreamento de satélites e predição orbital, com interface de rádio e rotor, desenvolvido por John A. Magliacane. É um software livre, os usuários podem redistribuir e/ou modificar sob os termos da General Public License (GPL). A figura 4 mostra a interface da aplicação. A figura 3 mostra sua interface.

Figura 3 – Interface do PREDICT.

Fonte: (MAGLIACANE, 2006)

O PREDICT fornece o rastreamento em tempo real de satélites e a informação da predição orbital para os usuários e aplicações de clientes nessas variedades:

- Pelo console do sistema;
- Pela linha de comando;
- Por um socket via rede;
- Pela geração de som via saída de áudio;

2.5.3 GPREDICT

Gpredict é uma interface gráfica desenvolvida em C/GTK para o predict. Dentre seus diferenciais se destacam a inexistência de um número limite de satélites e estações terrestres em uso, possibilidade de mostrar as informações em mapas, tabelas e diagramas, possibilidade de criação de módulos para uso simultâneo de multi-estações e multi-satélites. Interface para controle de rádio e rotores e interface gráfica configurável. A figura 4 mostra a interface do Gpredict.

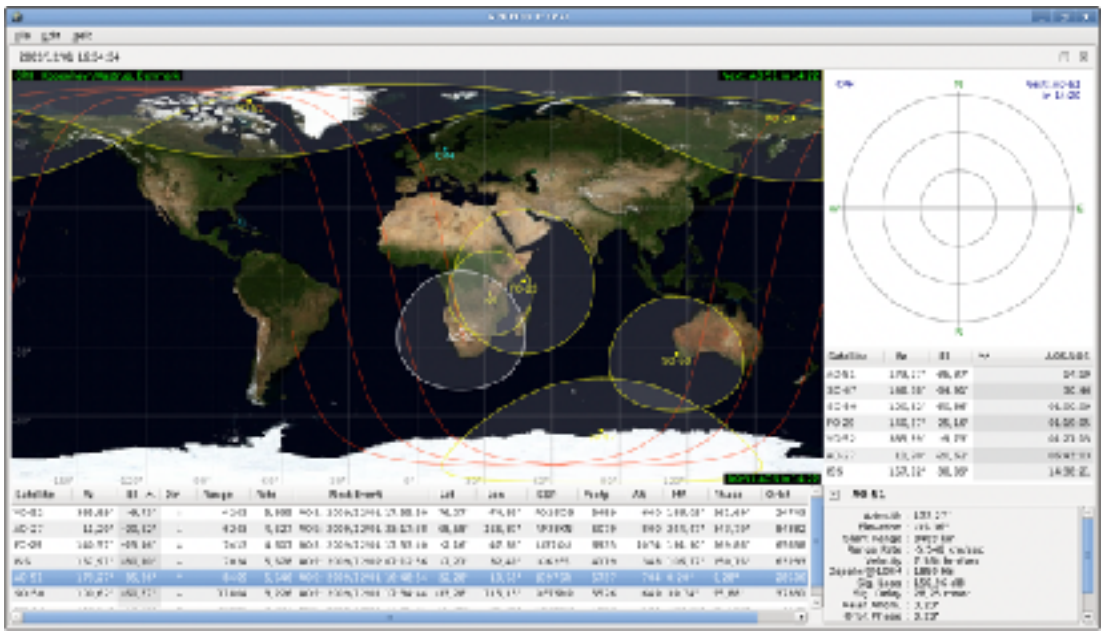


Figura 4 – Interface gráfica do Gpredict no LINUX.

Fonte: (CSETE, 2009)

2.5.4 Comparativo

Para operação standalone da RIBRAS, após uma comparação de funcionalidades, o sistema escolhido para interface com o operador da estação foi o Gpredict. Pois além de atender todas as necessidades, ele possui uma interface flexível e configurável, que permite apresentar apenas as informações necessárias para a operação. Além de ser uma aplicação de código aberto. A tabela 1 compara as funcionalidades dos softwares anteriormente descritos.

Tabela 1 – Comparativo de funcionalidades de softwares standalone.

Funcionalidade	Sistemas		
	Orbitron	Predict	Gpredict
Predição de passagem a partir de TLE	X	X	X
Rastreamento simultâneo	X	X	X
Tempo real / simulação	X		X
Interface gráfica	X		X
Interface flexível e configurável	X	X	X
Controle de rádio e rotor	X	X	X
Número ilimitado de satélites		X	X
Radar / Visão polar	X		X

2.6 Exemplos de Redes de Estações Terrestres

A fim de trazer o conhecimento de diferentes experiências, três exemplos de Redes de estações terrestres são introduzidos nestes capítulos, um considerado usar uma abordagem tradicional, com procedimentos básicos para troca de informações, e os outros dois, mais recentes e com base nos avanços mais recentes da Informática , sendo um considerado heterogêneo, ou feito por estações diferentes, e o outro é construído em cima do mesmo modelo de estação. Os próximos tópicos descrevem essas redes.

2.6.1 Abordagem de Estacão Terrestre Tradicional

Desde as primeiras missões espaciais, o segmento terrestre foi constituído de várias entidades, e a troca de informações entre essas partes pode ser considerada como rede. A agregação de diferentes estações já foi realizada no início da era espacial para obter uma melhor cobertura ou para aumentar a redundância. Um bom exemplo para esta abordagem "tradicional" ou "clássica" é o sistema ESA ESTRACK, composto por nove estações terrestres combinadas a uma rede, que já foi iniciada no início da década de 1970. A NASA também começou bastante cedo com a combinação de diferentes estações com redes, com a Deep Space Network (DSN), criada em 1958. (FISHER et al, 1999).

Antes de mencionar a abordagem da estação terrestre tradicional, um aspecto importante relacionado à taxonomia do termo estação terrestre precisa ser esclarecido: na literatura o segmento terrestre geralmente é dividido em controle de missão e rede de estações terrestres. O controle da missão é agregado no centro de controle (Centro de Controle de Missão (MCC), Centro de Controle de Operações de Aeronaves (SOCC)) e é responsável pelo planejamento de missão e operações de missão, por exemplo, monitorando e comandando a nave espacial. A rede da estação terrestre, composta por diferentes estações terrestres, está do outro lado tratando de recepção e transmissão de sinal, rastreamento de órbita, etc. O controle de missão e a rede da estação terrestre não são apenas divididos logicamente, também são geralmente separados geograficamente uns dos outros. Uma estação terrestre típica contém, neste contexto, várias estações receptoras, incluindo diferentes tipos de antenas, bem como equipamentos de hardware correspondentes. Por exemplo, a estação terrestre em Weilheim (parte do sistema ESTRACK) contém seis antenas diferentes que variam de 6 a 30 m para suportar missões espaciais e missões próximas da Terra. Assim, uma estação terrestre descreve, no sentido tradicional, um sistema sofisticado que contém um maior número de instalações para a comunicação por satélite, atuando como a interface entre satélite e centro de controle de missão. A rede de estação terrestre tradicional contém apenas algumas, porém, estações altamente especializadas, para suportar um amplo espectro de missões espaciais. Ao contrário, no contexto de pequenos projetos de satélites, uma estação terrestre é considerada como uma única entidade usada para acessar um único satélite, uma estação terrestre consiste aqui tipicamente apenas em um sistema de antena e equipamento correspondente para comunicação com um único satélite em LEO (MOCCIA et al., 2013).

O sistema ESTRACK é usado para explicar a organização geral de uma grande rede de estação terrestre. Mais especificamente, a arquitetura de uma estação terrestre típica, ou seja, uma estação de recepção de satélite será explicada. O núcleo da rede ESTRACK consiste atualmente em nove estações terrestres localizadas na Austrália, África, Europa e América do Sul. Desde 1968, a ESTRACK apoiou mais de 60 missões. As estações terrestres do sistema ESTRACK contêm muitas estações altamente especializadas. Assim, existem estações dedicadas para missões espaciais profundas e satélites geoestacionários e, em geral, não podem ser substituídas entre si. O sistema ESTRACK possui, por exemplo, antenas dedicadas de 35 m de diâmetro para missões em profundidade, e para missões próximas da Terra antenas de 15 m de diâmetro. As bandas de freqüência variam de S-Band a Ka-Band (MALDARI et al, 2008), portanto, todos os tipos de missões podem ser suportados pelo ESTRACK, mas as estações dedicadas para o tipo específico de missões precisam ser solicitadas.

Figura 5. Mapa mostrando as localizações das estações da ESA.

(ESA, Janeiro, 2016)

Uma estação terrestre exemplar para comunicação com um satélite LEO consiste tipicamente nas seguintes unidades: sistema de antena, transmissão e recepção, bem como equipamentos de telemetria, rastreamento e controle (TT & C): o sistema de antena contém aberturas de diferentes tamanhos, principalmente grandes antenas parabólicas para a recepção do sinal (o tamanho depende de vários fatores, o importante é fornecer o segmento, portanto, nem todos os satélites precisam ser operados ao mesmo tempo. Uma estratégia diferente é usar uma rede de estações terrestres altamente distribuídas para operar eficientemente uma distribuição distribuída segmento espacial. Um novo conceito de redes de estação terrestre de baixo custo evoluiu nos últimos anos, refletindo as abordagens de satélite distribuídas no segmento terrestre, também. É especialmente adequado para a operação de grande número de satélites. A vantagem dessas estações receptoras de baixo custo são que se assemelham à arquitetura e, portanto, são compatíveis com as plataformas de satélites mais pequenas. Para projeções satélites maiores, sistemas terrestres altamente distribuídos um novo passo nas operações de satélites. Os sistemas de antenas de estações de baixo custo foram projetados para bandas de freqüência específicas. O principal desafio é um conceito de rede inteligente para um segmento espacial distribuído em combinação com estações terrestres pouco acopladas.

Em contrapartida, a abordagem clássica da estação terrestre contém tipicamente um amplo espectro de hardware especializado, que limita a utilização a um tipo específico de missão ou tipo de satélite. Essas estações terrestres dedicadas não podem ser agrupadas arbitrariamente para a fase de operações de um sistema distribuído. Especialmente interessantes são os conceitos do campo de inteligência distribuída e controle para missões espaciais distribuídas, mas infelizmente, apenas alguns deles são transferíveis para o conceito de estação terrestre tradicional (MOCCIA et al., 2013).

2.6.2 Abordagem de Estação Terrestre Heterogênea

A Global Educational Network for Satellite Operations (GENSO) (PREINDL; PAGE; NIKOLAI- DIS, 2008) é formada por uma rede mundial de estações terrestres que interagem via padrão de software. O objetivo da GENSO é aumentar o retorno das missões espaciais

educacionais a mudar a forma com a qual as missões são gerenciadas aumentando drasticamente o nível de acesso a veículos espaciais orbitais educacionais.

Em novembro de 2007 foi feita uma demonstração no CubeSat Workshop. Uma apresentação em Aalborg, Dinamarca, descreveu para o resto do grupo da GENSO o trabalho desenvolvido na Universidade Politécnica da Califórnia. O cliente do controle da missão agendou sessões de downlink com a estação terrestre servidora. A mesma controla o rádio e os rotores da estação terrestre da AAU (Aalborg University).

O projeto GENSO foi iniciado sob a tutela do ISEB (International Space Education Board) e possuía várias habilidades existentes e o interesse de grandes grupos dentro e fora dos EUA. No entanto, ainda estava faltando uma estação terrestre totalmente equipada que pudesse complementar as suas capacidades. Desde 2011, um grande acordo de trabalho foi feito durante o lançamento 1E da GENSO para ajudar a fazer da rede mais amigável para o usuário e mais confiável para a comunidade de rádio amadores. Estações terrestres como a da COSMIAC (Universidade de Novo México), rodam vinte e quatro horas por dia, sete dias por semana, baixando dados para satélites educacionais como o veiculo espacial da Universidade de Michigan, o RAX-2.

O sistema da GENSO é um software em rede que permite ao usuário se comunicar com um veiculo espacial, usando uma estação terrestre remota que tenha uma visão limpa e clara do veiculo espacial. A comunicação entre o computador cliente (computador de controle da missão) e o servidor da estacão terrestre (o seu computador da estação) são feitos via Internet.

Um número de itens de hardware e de aplicações são necessárias para a GENSO funcionar. As tecnologias e frameworks que estão sendo desenvolvidos são detalhados a seguir:

- Servidores de Autenticação (AUS):

Esses servidores são equipados com o software da GENSO, que age como o núcleo central da rede. Eles não serão acessíveis pela grande maioria dos usuários, mas será usado como uma ferramenta de controle para os administradores rede.

Para evitar um ponto único de falha, existirão vários servidores de autenticação, conectados em diferentes backbones na internet que serão sincronizados caso necessário. Todas as instâncias do GSS e MCC, serão equipados para contatar o seu AUS mais perto como padrão, e então tentar os outros em caso de falha.

O conjunto dos servidores de autenticação, é conhecido coletivamente como Authentication Servers List (ASL). As tarefas da ASL são validar a identidade das instâncias do GSS e MCC quando eles logarem, manter em dia as listas dos atributos e estados de todas as estações terrestres e veículos espaciais da GENSO, distribuir essas listas de instâncias do GSS e MCC quando pedido e necessário, e servir a infraestrutura do projeto com dados de rede.

- Servidor da Estação Terrestre (GSS)

A aplicação GSS, quando logada no servidor de autenticação, ira permitir a participação das estações terrestres da GENSO, para automaticamente rastrear e estabelecer sessões de downlink com todos os veículos espaciais compatíveis - é claro que priorizando o veículo do proprietário, caso eles tenham algum.

A aplicação GSS também será usada para o upload de telecomandos para veículos espaciais compatíveis. Agendamento automatizado opcional e negociações com o MCC serão permitidas para missões particulares.

A aplicação será totalmente configurável e transparente para o usuário, e todos os dados poderão ser guardados localmente assim como podem ser transferidos para opera- dores espaciais.

- Cliente de Controle de Missão (MCC)

É a estação de controle para um Satélite. Cada satélite terá um MCC. A aplicação GSS ira ver um snapshot do satélite mas apenas o MCC terá a imagem inteira. Há também três componentes principais que o sistema da GENSO controla remotamente: Controlador dos Motores, TNC e Controlador do Rádio.

A GENSO é uma rede de cobertura global, suas estações estão localizadas na Europa, na África e nas Américas do Sul e do Norte. A figura 6 ilustra a cobertura da GENSO em Julho de 2012.

Figura 6 – Cobertura da rede GENSO em Julho de 2012.

Fonte: (GENSO, 2012)

2.6.3 Abordagem de Estacão Homogênea

A RIBRAS é uma iniciativa oficial da Agência Espacial Brasileira, destinado a apoiar missões nacionais e internacionais espaciais atuais e futuras. Cada estação terrestre da RIBRAS está equipada com duas torres independentes, uma para Banda-S e outra para comunicações UHF e VHF. Essas antenas são movidas por motores industriais robustos, capazes de fazer movimentos suaves, proporcionando que o que foi programado seja feito.
Ambas as antenas estão ligadas a um único rádio analógico-digital, que é um ICOM-9100, um rádio multi-banda com dois receptores independentes, ou seja, ele pode receber dois tipos de banda simultaneamente, e este foi comprado e é usado em parceria com a Tekever, uma empresa de equipamentos aeroespacial de Portugal, e a Universidade de São Paulo. Este rádio

pode ser controlado por um software em execução em um PC industrial que por sua vez faz a ligação lógica da estação para a rede. A automação da rede é feita através do Sistema da RIBRAS, que é um conjunto de softwares com o objetivo de sincronizar as estações terrestres através de um plano de trabalho gerado anteriormente e um conjunto de rotinas de controle, para se certificar de que a quantidade máxima de dados é coletada durante o sobrevoo dos satélites.

Em 2014 foi iniciado o projeto da RIBRAS, com a proposta de implementação de uma rede de estações terrestres para o rastreamento de satélites. As estações, para realizarem os rastreamentos de forma autônoma, recebem seus jobs (tarefas) com os satélites e suas passadas. O inicio do desenvolvimento do software para a automação e orquestração das estações foi esboçado. O lado do Cliente da estação (que consiste em job controller e o antenna controller), foi desenvolvido, porem é necessário um operador em frente ao pc industrial no local da estação terrestre para realizar o rastreamento do satélite. Nesse mesmo ano, foi apresentado no 1st IAA Latin America CubeSat Workshop, realizado em Brasília, um artigo com o título de: A arquitetura do software para a rede de estações terrestres da RIBRAS. O artigo detalha a arquitetura do software da rede de estações terrestres e demonstra como os softwares iriam interagir e sua integração para a rede se tornar autônoma.

Já no ano de 2015, foi implantada a estação terrestre na UPEA, sendo montada e instalada pelos membros do CRSEA. Com a estarão terrestre em funcionamento o software do cliente foi instalado no PC industrial para a realização de rastreamentos dos satélites NOAAs e SERPENS. Em 2016, no congresso do 2nd IAA Latin American CubeSat Workshop, realizado em Florianópolis, foram apresentados 2 artigos, o primeiro com o título de: Desenvolvendo e operando uma estação terrestre automatizada, e o segundo, com o título de: Orquestração e Controle de uma rede de estações terrestres automatizadas. Em 2017, foi apresentado o artigo A Nation-Wide Ground Station Network: the RIBRAS Project, no 1st IAA Latin American Symposium on Small Satellites, em Buenos Aires, Argentina.

3 METODOLOGIAS DE DESENVOLVIMENTO

A pesquisa será desenvolvida em cima da estação terrestre já existente, situada no PICG. O software que será desenvolvido para o auxilio da automação da rede de estações será feito na linguagem de programação python. Para a interface com o operador, o cliente standalone utilizará o software RIBRASPredict, que foi criado pelo CRSEA.

3.1 Arquitetura da RIBRAS

O sistema RIBRAS (Rede Integrada Brasileira de Rastreamento de Satélite) é um conjunto de softwares com o objetivo de controlar a rede de estações terrestres que receberá os dados de satélites brasileiros. As estações terrestres trabalham de forma sincronizada, seguindo um plano de trabalho gerado anteriormente, para se certificar de que a quantidade máxima de dados são coletados durante sobrevoos dos satélites (CARVALHO, R. A. et al., 2014). O Software está organizado em um modelo de comunicação master/slave. Cada um dos pacotes, Servidor (master) e Cliente (slave), tem três módulos: Scheduler, Distributor e Controller para o servidor e Job Controller, Antenna Controller e Communication Controller para cada cliente - as estações terrestres. A figura 7 ilustra a organização destes pacotes.

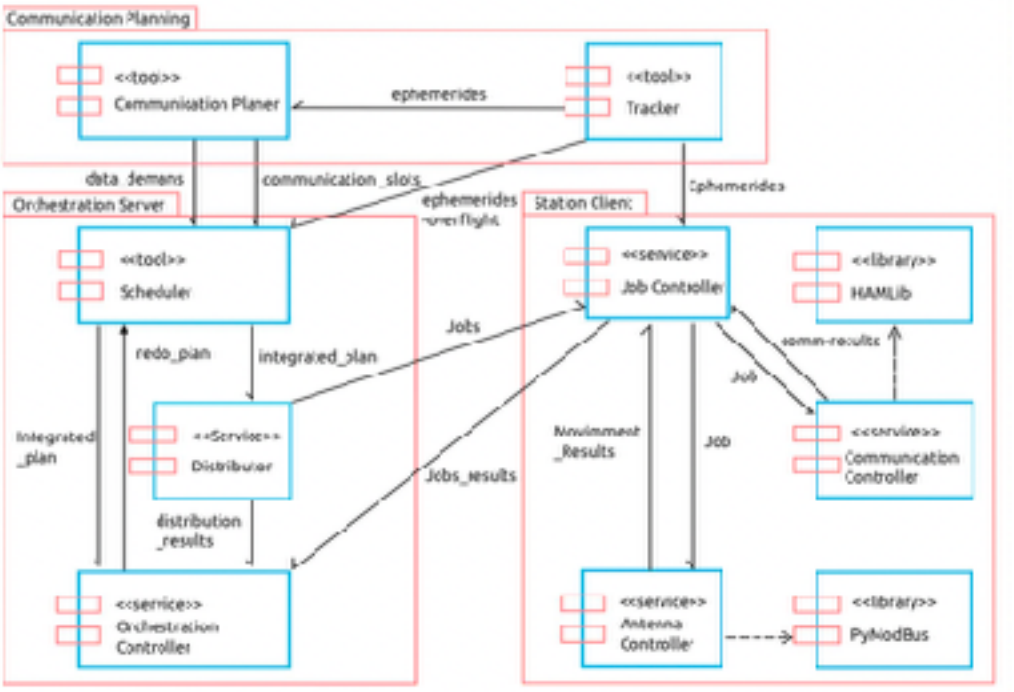


Figura 7 – Estrutura de pacotes da RIBRAS.

Fonte: Autor (2014)

A parte da orquestração e da implementação dos estágios necessários para a automação da estação serão calculados matematicamente e só então aplicados no software. Serão realizadas três etapas: planejamento, programação e controle.

No lado master, o Scheduler é responsável pela geração de um plano integrado para rastrear os satélites de uma determinada constelação durante um determinado período de tempo - normalmente, o tempo entre o primeiro e o último satélite. O Distributor divide o plano em tarefas específicas que serão enviadas a cada estação correspondente. Finalmente, o Orchestration Controller fará o controle da execução dessas tarefas criadas pelo Distributor, recebendo o feedback das estações e analisando as informações recebidas, como erros físicos e de software. Caso exista algum erro de rastreamento ou outro qualquer na rede, o controller envia esses erros para o Scheduler para a analise sensitiva do plano integrado. Caso seja necessário e haja tempo, um novo plano integrado é gerado, reiniciando o ciclo inteiro (HISSA, L. R. et. al, 2016).

Em cada estação - o lado slave - o Job Controller recebe os jobs, prepara a sua execução e, no momento certo, os executa. Durante a execução do trabalho, o Antenna Controller e Communication Controller controlam, respectivamente, a posição da antena e do funcionamento de rádio.

A RIBRAS tem dois modos de operação: um modo automático e um modo manual.

Modo Manual:

Permite o uso das estações terrestres para operadores locais, usando um software chamado de RIBRASPredict, que foi customizado a partir do GPredict para atender satisfatoriamente as necessidades dos operadores, e será explicado no próximo tópico. Os operadores deverão manter uma rotina, integrada com as operações automáticas.

Em modo de operação Standalone o usuário que estiver operando a estação terrestre irá usar o RIBRASpredict para escolher os satélites que desejar rastrear. Nesse modo o primeiro módulo a ser usado é o gpredict controller que abre dois sockets para se comunicar com a interface do RIBRASPredict um para os dados de posicionamento de antena que serão enviados para o Antenna Controller, que será responsável pelo controle autônomo dos rotores

e outro para os dados do rádio que serão enviados para o Communication Controller, que será responsável pelo controle autônomo do rádio.

Modo Automático:

É nesse modo que este trabalho foca sua automatização. Fazer com que todas as estações se comuniquem e façam seus rastreios no modo automático.Permite o uso das estações sem nenhuma operação local, de maneira 100% autônoma.

Pelo lado do servidor:

- O plano integrado é gerado, quebrado em partes específicas de cada estação, chamados de jobs, e distribuído para as estações responsáveis por cada parte.
- Após a execução das tarefas ele analisa os resultados e os armazena.

Um plano integrado contém todas as informações das próximas passagens a serem rastreadas pela estação terrestre local, junto com as TLEs (NASA, 2011) dos satélites que estarão nessas passagens.

3.2 Desenvolvimento

O foco deste trabalho é o desenvolvimento do módulo chamado de Distributor do lado servidor do software. Para o desenvolvimento do mesmo, foi utilizada a linguagem de programação python, orientada a objeto, com uma bateria de testes em TDD e objetos Mock.

Com a separação do plano integrado que é recebida do Scheduler, os jobs são enviados para o Job Controller de cada cliente em cada estação. Com os resultados distribuídos, o Distributor envia-os para o Orchestration Controller, que com os resultados do Job Controller e o plano integrado do Scheduler, irá avaliar a necessidade de uma reprogramação do plano integrado, e caso seja necessário, irá informar ao Scheduler para que seja feito um novo plano.

A implementação do Distributor foi feita por etapas, sempre acompanhadas com testes. Primeiramente, foi implementado o método de separação de jobs, que recebe o plano e o separa de acordo com as estações clientes. Em outro método, é feito o envio dos jobs que

foram separados no método anterior. Juntamente com o desenvolvimento do Distributor, foram feitos os testes das funcionalidades dos seus métodos.

Primeiramente, é feita a configuração (setup) do distributor e de um exemplo de plano integrado, como visto no exemplo de código 1. O setup cria como exemplos, uma instância da classe Distributor, um plano integrado recebido do Scheduler, e um arquivo de configuração, que informa o endereço e porta de qual estacão está sendo enviado os dados e de onde vem esse trabalho para a estacão.

```
def setUp(self):
    self.distributor = Distributor()
    self.plan = [
                  12569537329,
                  {
                    '14bisat':'1 25544U 98067A   08264.51782528 -.00002182  00000-0 -
                    'sat2':'1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11
                  },
                  [
                    {'station':'Brasilia', 'sat':'14bisat', 'it':1000, 'ft': 1200},
                    {'station':'Brasilia', 'sat':'sat2', 'it':2200, 'ft': 2400}
                  ]
                ]
    #setting configuration file
    self.config_file = StringIO('[Brasilia]\naddress: 0.0.0.1\nport: 8000\n\n[Campos]
    Station.Config.set_config_file(self.config_file)
```

Código 1 – Exemplo da configuração para teste do Distributor.

Fonte: Autor (2018)

Porém, para a aplicação dos testes, foram necessários Mocks de dois módulos que recebem dados do Distributor: O Job Controller e o Orchestration Controller. Os códigos 2 e 3 demonstram a criação dos Mocks, que simulam o recebimento dos parâmetros do Job Controller e do Orchestration Controller e as respostas esperadas.

```
def it_should_communicates_with_Job_Controller(self):
    self.jobcontrollermock = JobController()
    self.jobcontrollermock.load_job = MagicMock(return_value=True)
    jobMock = self.jobcontrollermock.load_job(True)
    jobMock |should| be(True)
```

Código 2 – Mock do Job Controller.

Fonte: Autor (2018)

```
def it_should_communicates_with_Orchestration_Controller(self):
    self.orchestrationmock = OrchestrationController()
    self.orchestrationmock.redo_plan = MagicMock(return_value=True)
    orchMock = self.orchestrationmock.redo_plan(True,True, self.plan)
    orchMock |should| be(True)
```

Código 3 – Mock do Orchestration Controller.

Fonte: Autor (2018)

Após o Mock ter sido bem sucedido, é feito um teste de funcionalidade no método de separação do Distributor que se chama *_separate_jobs*. Este é o método que recebe o plano integrado do módulo Scheduler, e o separa em jobs. No código 4, é demonstrado o teste em TDD deste método.

```
def it_separates_jobs(self):
    jobs = self.distributor._separate_jobs(self.plan)
    Brasilia_job = jobs[0]
    Brasilia_job['station'] |should| equal_to('Brasilia')
    Brasilia_job['gz'] |should| equal_to(12569537329)
    Brasilia_job['tles'] |should| equal_to({
                                    '14bisat':'1 25544U 98067A   03264.51782528 -.000
                                    'sat2':'1 25544U 98067A   08264.51782528 -.000021
                                  })
    Brasilia_job['tasks'] |should| equal_to([
                                     {'sat':'14bisat', 'it':1600, 'ft': 1200},
                                     {'sat':'sat2', 'it':2200, 'ft': 2400}
                                   ])
```

Código 4 – Teste do método *_separate_jobs*.

Fonte: Autor (2018)

O teste foi escrito para verificar e separar cada parte do plano, que vem no seguinte formato:

- Local da estacão escolhida;
- Número da estacão terrestre;
- TLEs que são de estacão escolhida;
- Tarefas da estacão escolhida;

Com a separação efetuada, é verificado no teste se o que o distributor separou realmente é corretamente o esperado.

Os próximos testes foram aplicados separadamente para verificar e esperar uma falha na comunicação do Distributor entre o Job Controller e o Orchestration Controller. A falha acontece quando o método *distribute_jobs,* que é responsável pela distribuição dos jobs para os Jobs Controllers de cada estação, e para o Orchestration Controller, por algum motivo, não efetua a distribuição adequadamente, seja por algum problema no lado do servidor quanto do lado do cliente.

Esse teste é feito para verificar como o módulo do software reage mesmo em casos de falha, e se a reação é a esperada. Para simular a falha, foi usado o Mock do Job Controller e do Orchestration Controller, porém com a diferença de, ao invés de enviar uma resposta de que não houve problemas na comunicação, é simulada uma resposta de falha no momento em que o Job Controller e o Orchestration Controller recebem os jobs. Um exemplo de cada falha pode ser visto nos códigos 5 e 6, respectivamente.

```
def it_should_fail_sending_to_Job_Controller(self):
    self.jobcontrollermock = JobController()
    self.jobcontrollermock.load_job = MagicMock(return_value=False)
    jobMock = self.jobcontrollermock.load_job(True)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(1)
    jobMock = self.jobcontrollermock.load_job(job_distribution)
    jobMock |should| be(False)
```

Código 5 – Teste de falha no envio para o Job Controller.

Fonte: Autor (2018)

```
def it_should_fail_sending_to_Orchestration_Controller(self):
    self.orchestrationmock = OrchestrationController()
    self.orchestrationmock.redo_plan = MagicMock(return_value=False)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(2)
    orchMock = self.orchestrationmock.redo_plan(False,True, self.plan)
    orchMock |should| be(False)
```

Código 6 – Teste de falha no envio para o Orchestration Controller.

Fonte: Autor (2018)

Após os testes de falhas bem sucedidos separadamente, também é testado caso tenha falha na comunicação dos dois Controllers ao mesmo tempo. O código 7 demonstra o teste.

```
def it_should_fail_sending_to_all(self):
    self.jobcontrollermock = JobController()
    self.jobcontrollermock.load_job = MagicMock(return_value=False)
    self.orchestrationmock = OrchestrationController()
    self.orchestrationmock.redo_plan = MagicMock(return_value=False)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(3)
    jobMock = self.jobcontrollermock.load_job(job_distribution)
    jobMock |should| be(False)
    orchMock = self.orchestrationmock.redo_plan(True,False, self.plan)
    orchMock |should| be(False)
```

Código 7 – Teste de falha no envio para os Job Controller e Orchestration Controller.

Fonte: Autor (2018)

Assim como os testes de falha, foram feitos testes de comunicação bem sucedida entre o Distributor e os Controllers. A diferença entre estes é no objeto Mock, que ao invés de retornar um valor de falha, ou seja, um valor *False,* ele retorna um valor de sucesso, um *True.* Assim, o método distribute_jobs envia os jobs e é informado com uma resposta de que a comunicação foi bem sucedida. Também foram feitos testes com os controllers separadamente, vistos nos códigos 8 e 9.

```
def it_should_send_jobs_to_Job_Controller(self):
    self.jobcontrollermock = JobController()
    self.jobcontrollermock.load_job = MagicMock(return_value=True)
    jobMock = self.jobcontrollermock.load_job(True)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(4)
    jobMock = self.jobcontrollermock.load_job(job_distribution)
    jobMock |should| be(True)
```

Código 8 – Teste de comunicação bem sucedida com o Job Controller.

Fonte: Autor (2018)

```
def it_should_send_jobs_to_Orchestration_Controller(self):
    self.orchestrationmock = OrchestrationController()
    self.orchestrationmock.redo_plan = MagicMock(return_value=True)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(5)
    orchMock = self.orchestrationmock.redo_plan(True,True, self.plan)
    orchMock |should| be(True)
```

Código 9 – Teste de comunicação bem sucedida com o Orchestration Controller.

Fonte: Autor (2018)

Já o código 10, demonstra o último teste feito, que aplica a distribuição de jobs simultaneamente para o Job Controller e Orchestration Controller, esperando e recebendo uma resposta de sucesso.

```
def it_should_send_jobs_to_all(self):
    self.jobcontrollermock = JobController()
    self.jobcontrollermock.load_job = MagicMock(return_value=True)
    self.orchestrationmock = OrchestrationController()
    self.orchestrationmock.redo_plan = MagicMock(return_value=True)
    jobs = self.distributor._separate_jobs(self.plan)
    job_distribution = self.distributor.distribute_jobs(jobs)
    len(BUFFER) |should| be(6)
    jobMock = self.jobcontrollermock.load_job(job_distribution)
    jobMock |should| be(True)
    orchMock = self.orchestrationmock.redo_plan(True,True, self.plan)
    orchMock |should| be(True)
```

Código 10 – Teste de comunicação bem sucedida com o Job Controller e o Orchestration Controller.

Fonte: Autor (2018)

4 RESULTADOS

Os resultados obtidos foram positivos, cumprindo o objetivo deste trabalho. Porém, para alcançar o sucesso obtido os testes passaram por falhas e casos em que não estavam passando corretamente. A seguir será mostrada as baterias de testes sendo rodadas e tendo eventuais falhas ao longo do processo deste trabalho. Quando um teste é bem sucedido, ele é mostrado em verde no prompt de comando, e caso haja alguma falha no teste, ele é mostrado em vermelho.

Primeiramente, o início dos testes não tiveram dificuldades ou problemas. A figura 8 mostra o resultado do primeiro teste, que foi escrito de acordo com o plano integrado enviado pelo Scheduler e verifica se a separação desse plano foi feita corretamente pelo Distributor.

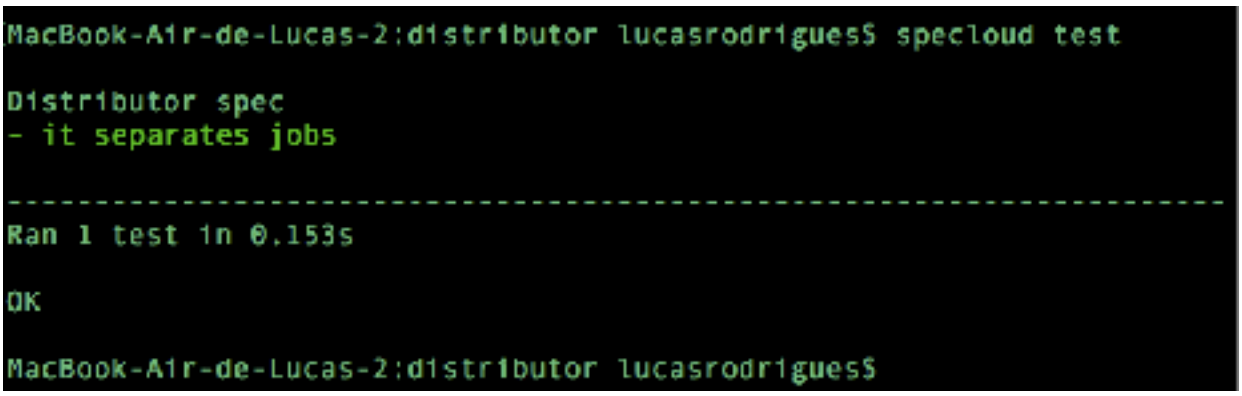


Figura 8 – Resultado do teste de separação dos jobs.

Fonte: Autor (2018)

Em seguida, foram necessários os testes para avaliar se os objetos Mock feitos estavam respondendo conforme o esperado. Não houveram falhas na criação dos objetos, como demonstra a figura 9.

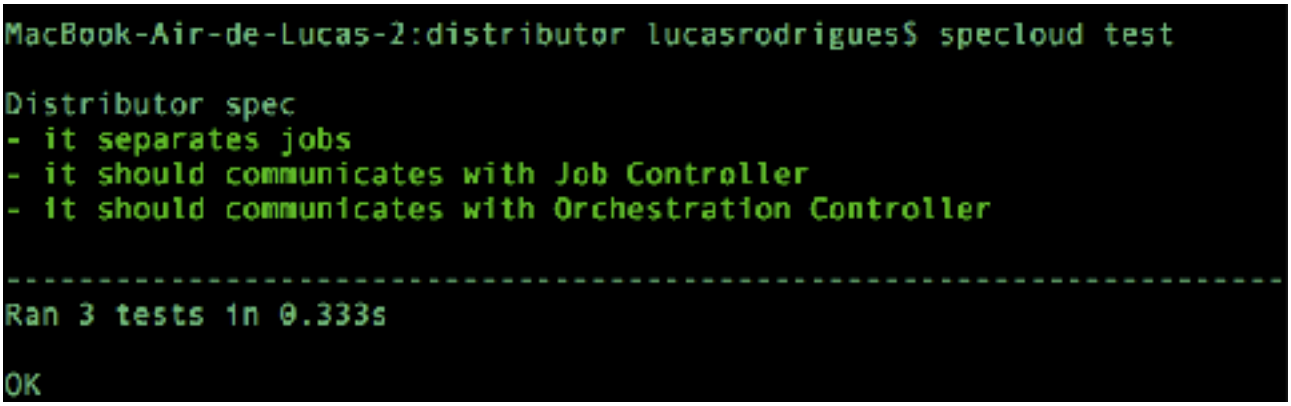


Figura 9 – Resultado dos testes de objetos Mock dos Controllers.

Fonte: Autor (2018)

Com os testes dos objetos Mocks passando com sucesso, o proximo passo foi rodar a bateria de testes para avaliar se os testes de falhas de comunicação com o Job Controller e o Orchestration Controller estavam passando. Porém, como mostra a figura 10, apenas um dos testes passou, demonstrando uma falha no outro teste.

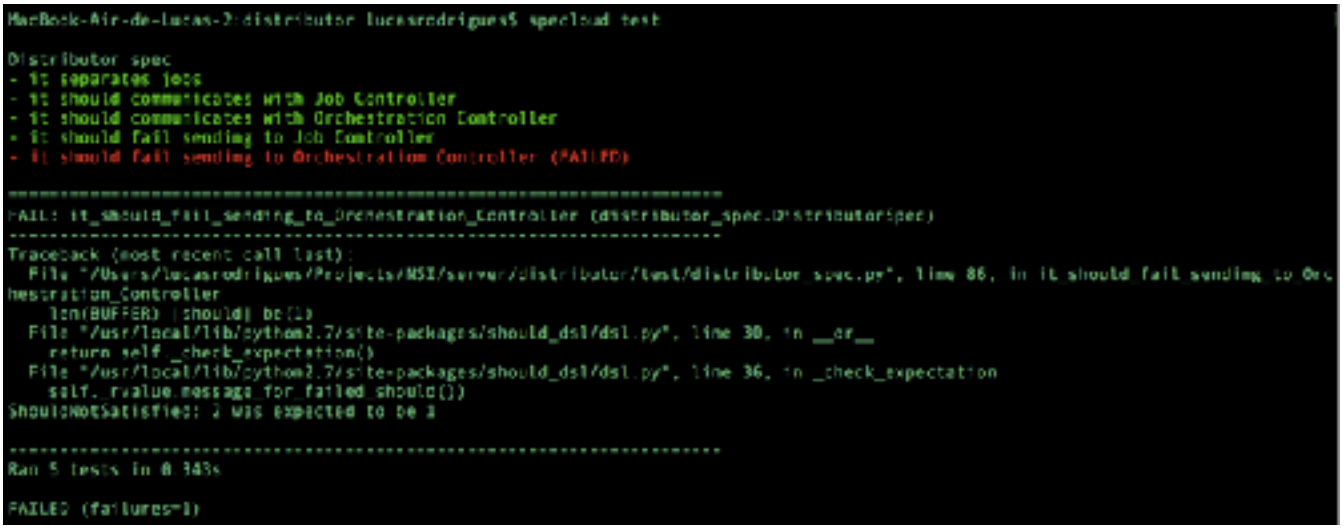


Figura 10 – Falha no teste de falha de comunicação do Orchestration Controller.

Fonte: Autor (2018)

O erro demonstrado era um problema no buffer do Distributor, que armazena o número de vezes em que se estabelece uma comunicação e uma distribuição dos jobs. Com a falha localizada e descoberta, foi alterado o tamanho esperado do buffer e o teste roda com sucesso, como na figura 11.

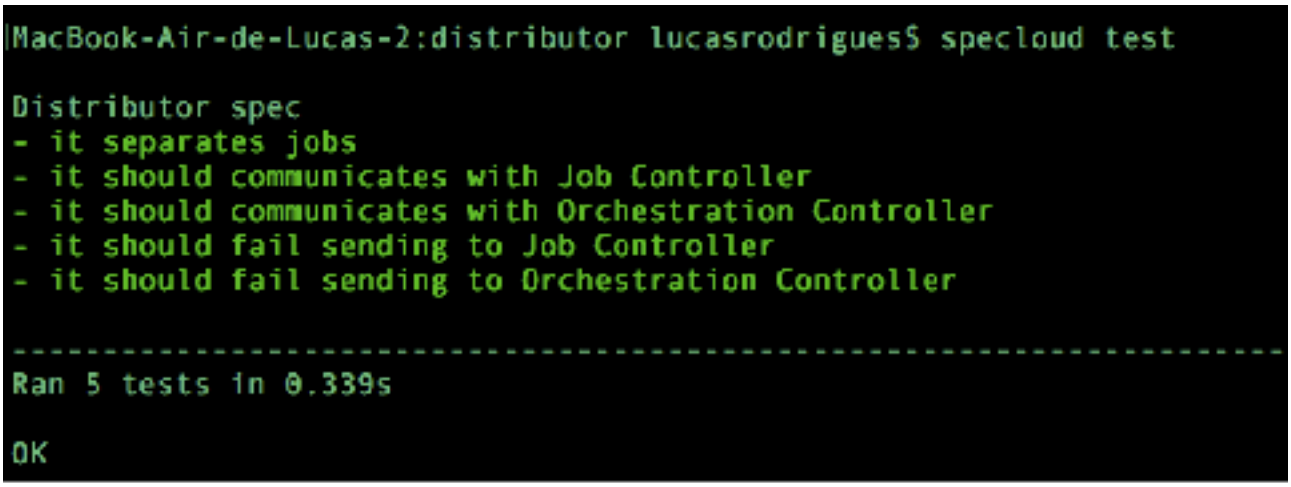


Figura 11 – Teste de falha de comunicação do Orchestration Controller passando.

Fonte: Autor (2018)

O próximo passo para os testes do Distributor foram os testes de sucesso nas comunicações entre os Job Controller e Orchestration Controller. Nestes testes, foram testados Com os testes escritos, o passo seguinte foi a execução dos mesmos, resultando nas falhas de ambos, mostrado na figura 12.

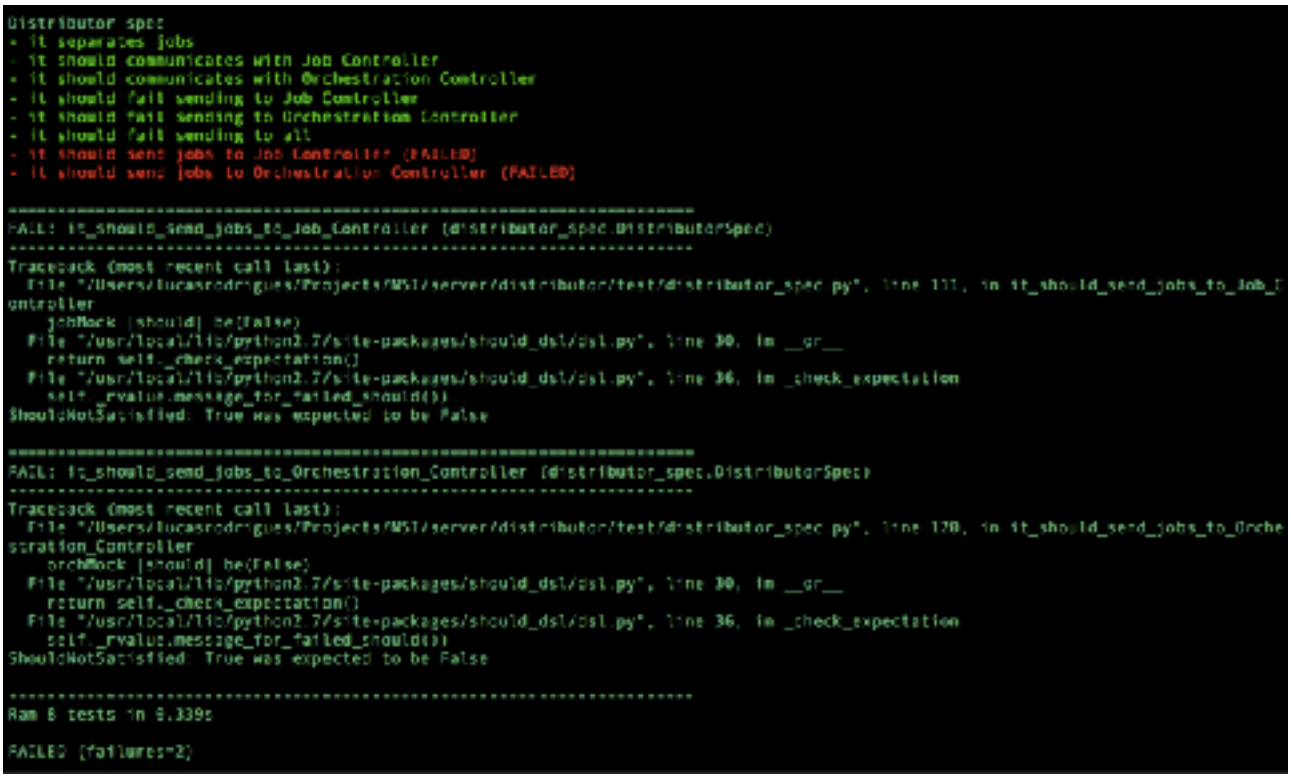


Figura 12 – Falha nos testes para comunicação.

Fonte: Autor (2018)

As falhas ocorreram pois a confirmação de que a comunicação veio como *False,* indicando que durante o teste a mesma tinha falhado. O esperado era que a confirmação viesse como *True,* assim tendo o teste efetuado sua verificação e passado sem falha alguma.

Com algumas modificações e ajustes no código do Distributor, que estava interpretando de maneira equivocada as confirmações, os testes foram rodados novamente, e dessa vez, como na figura 13, todos os resultados dos testes foram como esperados.

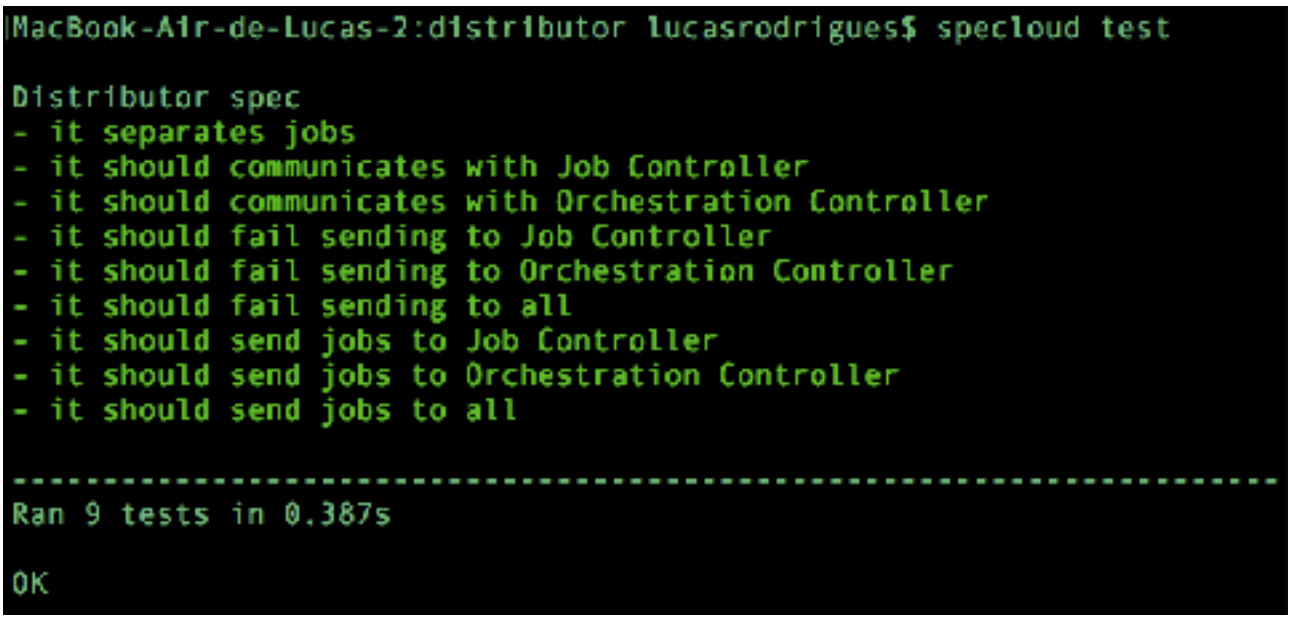


Figura 13 – Testes do Distributor passando como esperado.

Fonte: Autor (2018)

O desenvolvimento e implementação do Job Controller já estão bastante avançados, já com funcionalidades e testes sendo implementados. Porém, o Orchestration Controller encontra-se em sua fase inicial de desenvolvimento e implementação, sendo assim, o uso de objetos Mock a melhor escolha para demonstrar a funcionalidade e integração do módulo Distributor no lado servidor do software da RIBRAS.

Ao final do processo de implementação e desenvolvimento do Distributor, foram escritos 9 testes para verificar as funcionalidades e os dados do módulo, para que o mesmo opere com total segurança e produtividade.

Com todos os testes conferidos e passando sem adversidades, o desenvolvimento do módulo Distributor no software da RIBRAS pode ser considerado funcional e implementado com sucesso, pronto para sua inclusão e adaptação no código do software do lado servidor da RIBRAS.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Como uma rede de estações terrestres como a RIBRAS é algo inédito no Brasil. existem várias adversidades a serem superadas, como por exemplo, a implementação do software nas estações terrestres e o desenvolvimento dos outros módulos, tanto do lado do cliente quanto do lado do servidor, para o desenvolvimento completo do software autônomo da RIBRAS.

Mesmo com tais adversidades, o desenvolvimento do software ainda está sendo feito para os trabalhos futuros, tanto os módulos do lado cliente, como o Job Controller, quanto os módulos do lado servidor, como o Orchestration Controller.

Estima-se que o desenvolvimento desta dissertação irá contribuir de forma significativa para o rastreamento de satélites na região onde estão situadas as estações terrestres, além da criação de um software que possibilita a utilização das estações sem a necessidade de uma mão de obra qualificada, e também a redução de erros humanos.

REFERÊNCIAS BIBLIOGRÁFICAS

AMADURO, L. R.; CARVALHO, R. A. ; HISSA, L. R. ; VIANNA, W. S. ; OLIVEIRA, W. S. ; SOUZA, S. M. ; MOURA, L. G. . **Developing and Operating an Automated Ground Station**. In: Second IAA Latin American CubeSat Workshop, 2016, Florianópolis. Annals of the Second IAA Latin American CubeSat Workshop, 2016. p. 2-3.

BAKER, K.; JANSSON, D. **Space satellites from the world's garage - the story of amsat.** In: Proceedings of the IEEE National Aerospaceand Electronics Conference. [S.l.: s.n.], 1994. p. 1174–1181.

BRIEß, K.; SCHILLING, K. A**nalyse der Anwendungsfelder und des Nutzungspotentials von Pico- und Nano-Satelliten: executive summary.** 2008. Disponível em: <http: // books.google.com.br/books?id=rnYBSQAACAAJ>.

CAKAJ, S.; KEIM, W.; MALARIC, K. **Communications duration with low earth orbiting satellites.** In: Proceedings of the 4th IASTED International Conference on Antennas, Radar and Wave Propagation. [S.l.: s.n.], 2007.

CARVALHO, R. A. ; AMADURO, L. R. ; HISSA, L. R. ; MOURA, L. G. ; CORDEIRO, C. S. ; LISBOA, W. S. . **The Software Architecture for the RIBRAS Ground Station Network.** In: 1st IAA Latin America CubeSat Workshop, 2014, Brasília. Proceedings of the 1st IAA Latin America CubeSat Workshop, 2014. p. 1-2.

DAGANZO, E.; FELTHAM, S.; MUCH, R.; NASCA, R.; STALFORD, R.; HESS, M.P.; STRINGHETTI, L.; CACCIAPUOTI, L. **ACES ground segment functionality and preliminary operational concept.** 2009 IEEE International Frequency Control Symposium

Joint with the 22nd European Frequency and Time Forum; Besancon; France; 20 April 2009 through 24 April 2009.

DAVID, P.; HITEFIELD, S.; LEFFKE, Z.; HEADLEY, W.C.; MCGWIER, R.W. .I**mplementation of an actor framework for a ground station.** 2016 IEEE Aerospace Conference, AERO 2016; Big Sky; United States; 5 March 2016 through 12 March 2016.

HAEUSLER, M.; WIEDEMANN, K. Bodenstationsnetzwerk. **In: Hand-buch der Raumfahrttechnik.** [S.l.]: Hanser, 2008. p. 464–483.

HEIDT, M. H. et al. **Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation.** In: 14TH Annual/USU Conference on Small Satellites. [s.n.], 2000. Disponível em: <http://digitalcommons.usu.edu/cgi/viewcontent. cgi? article=2069&context=smallsat>.

HESS, W. **The Radiation Belt and Magnetosphere.** [S.l.]: Blaisdell, 1968.

HISSA, L. R. ; AMADURO, L. R. ; CARVALHO, R. A. ; MOURA, L. G. . **Orchestration and Controlling of an Automated Ground Station Network.** In: Second IAA Latin American CubeSat Workshop, 2016, Florianópolis. Annals of the Second IAA Latin American CubeSat Workshop, 2016. p. 4-5.

HITEFIELD, S.; LEFFKE, Z.; FOWLER, M.; MCGWIER, R.W..**System overview of the Virginia Tech Ground Station.** 2016 IEEE Aerospace Conference, AERO 2016; Big Sky; United States; 5 March 2016 through 12 March 2016.

KARN, P.; PRICE, H.; DIERSING, R. **Packet radio in the amateur service.** IEEE Journal on Selected Areas in Communications, v. 3, n. 3, p. 431–439, 1985.

LIMA, L. F. **O Crm No Contexto Das Bibliotecas: Proposta De Implementação Na Biblioteca Central Do Cefet/Rj.** In Universidade Federal Fluminense, Departamento De Engenharia De Produção, Laboratório De Tecnologia, Gestão De Negócios E Meio Ambiente, Mestrado Profissional Em Sistemas De Gestão, 2013, Niterói.

MAGLIACANE, J. A. **PREDICT - A Satellite Tracking/Orbital Prediction Program.** 2006. Disponível em: <http://www.qsl.net/kd2bd/predict.html>.

NASA. **Definition of Two-line Element Set Coordinate System.** 2011. Disponível em: <http://space ight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSSOP/ SSOP_Help/ tle_def.html>.

PREINDL, B.; PAGE, H.; NIKOLAIDIS, V. **Genso: The global educational network for satellite operations.** In: Proceedings of the 59th International Astronautical Conference. [S.l.: s.n.], 2008.

PREINDL, B. et al. **Measuring satellite link quality in ground station networks with heterogenous hardware environments.** In: Vienna University of Technology and Aalborg University, Denmark, Tech. Rep. [S.l.: s.n.], 2008.

ROYSAM, B. et al. E**levating the ECE Capstone Design Experience with CubeSats.** 2014. Disponível em: <http://ecedha.org/ece-media/newsletter/january-2014/ cubesats-article? utm_source=ECE+Source+-+January+2014+-+Final+1-9-14& utm_campaign=ECEDHA+Test+Newsletter&utm_medium=email>.

SCHMIDT, M. **Ground Station Networks for Efficient Operation of Distributed Small Satellite Systems.** Tese (Doutorado) — Universität Würzburg, Fakultät für Mathematik und Informatik, 2011.

TULI, T. S.; ORR, N. G.; ZEE, R. E. **Low cost ground station design for nanosatellite missions.** In: Proceedings of AMSAT Symposium. [S.l.: s.n.], 2006.

WAYDO, S.; HENRY, D.; CAMPBEL, M. C**ubesat design for leo-based earth science missions.** Aerospace Conference Proceedings. IEEE, v. 1, p. 435–455, 2002.

WERTZ, J. R.; LARSON, W. J. **Space Mission Analysis and Design (SMAD).** 3. ed. [S.l.]: Space Technology Library, 2005.

BECK, K. **Test-Driven Development by Example.** Addison Wesley - Vaseem, 2003.

FREEMAN, S.; PRYCE, N. **Growing Object-Oriented Software Guided by Tests.** 1st ed. 2009.

FISHER, F., MUTZ D, ESTLIN T, PAAL L, LAW E, GOLSHAN N, CHIEN S. **The past, present, and future of ground station automation within the DSN.** In: Proceedings of the 1999 I.E. aerospace conference, Aspen, CO, pp 315–324, 1999.

MALDARI, P., BOBRINSKIY N. **Cost efficient evolution of the ESA network in the space era.** Space Oper Commun 5:10–18, 2008.

MOCCIA, A. ; RENGA, A ; D'ERRICO, M. . **Distributed Space Missions for Earth System Monitoring,** SPACE TECHNOLOGY LIBRARY. Published jointly by Microcosm Press and Springer, 2013.

Anexos

BIBLIOMETRIA

O levantamento bibliográfico foi adaptado da bibliometria feita por Lima (2013) compreendeu a realização das seguintes estratégias:

- Busca direta em catálogos de bibliotecas e na Internet, em mecanismos de busca como Google e Google Acadêmico;
- Consulta à base Scopus, do Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES); e
- Consulta a revistas e a artigos científicos disponíveis na Scientific *Electronic Library Online* (SciELO).

O levantamento bibliográfico foi feito por meio da análise bibliométrica dos materiais encontrados nas buscas. A Tabela 1 apresenta todas as palavras e termos utilizados para a realização do levantamento bibliográfico, incluindo as associações feitas com a utilização dos operadores booleanos E e OU. Já a Figura 14 mostra como se deu o fluxo da análise bibliográfica realizada, tanto no levantamento realizado na base Scopus, quanto os obtidos em consultas a catálogos de bibliotecas, na Internet, em banco de teses e dissertações e na SciELO.

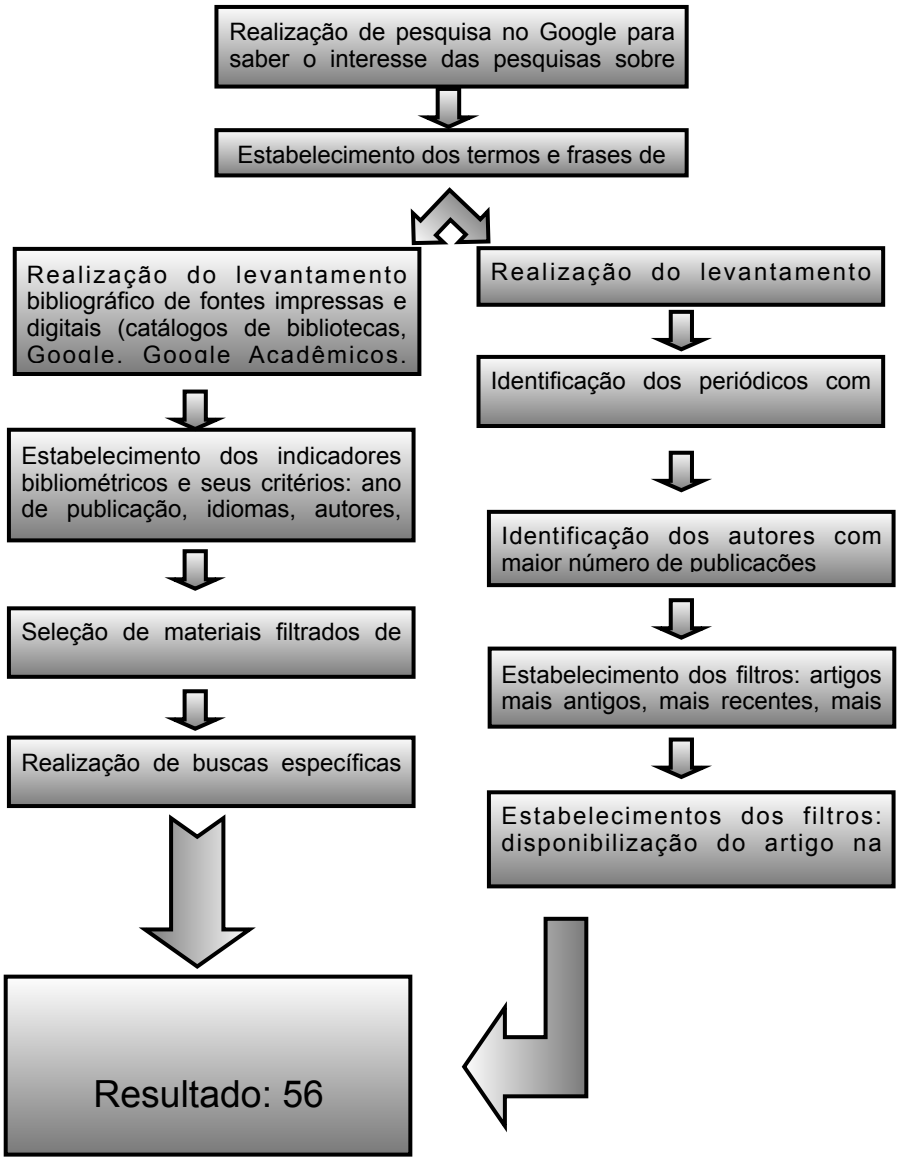
A pesquisa foi feita com as seguintes palavras-chave e seus tesauros: “Ground Segment”, “Distributed Systems” e “Aerospace”.

Tabela 2 –Pesquisa nas bases e revistas.

Termos	Pesquisa	Ocorrências
A	TITLE-ABS-KEY(Ground AND Segment) OR TITLE-ABS-KEY(Ground AND Station)	40.230
B	TITLE-ABS-KEY(Distributed AND Systems)	426.860
C	TITLE-ABS-KEY(Aerospace)	116.180
A & B	TITLE-ABS-KEY(Ground AND Segment) OR TITLE-ABS-KEY(Ground AND Station) AND TITLE-ABS-KEY(Distributed AND Systems)	941
B & C	TITLE-ABS-KEY(Distributed AND Systems) AND TITLE-ABS-KEY(Aerospace)	1.737
A & C	TITLE-ABS-KEY(Ground AND Segment) OR TITLE-ABS-KEY(Ground AND Station) AND TITLE-ABS-KEY(Aerospace)	1.260
A & B & C	TITLE-ABS-KEY(Ground AND Segment) OR TITLE-ABS-KEY(Ground AND Station) AND TITLE-ABS-KEY(Distributed AND Systems) AND TITLE-ABS-KEY(Aerospace)	56

Figura 14: Fluxo da análise bibliográfica.

Fonte: Autor (2017), adaptado de Lima (2013)



Foram contemplados no referencial teórico deste estudo os seguintes tipos de publicações: trabalhos acadêmicos de conclusão de curso (TCC), dissertações, teses, trabalhos apresentados em eventos, artigos, livros e capítulos de livros. Assim sendo, conforme apresenta o Gráfico 1, a maior parte do material consultado é referente a artigos, totalizando 63% do referencial total. O segundo tipo de material mais consultado são os trabalhos apresentados em eventos, com 15% do referencial total. Capítulos de livros e livros completos seguem, respectivamente, com 12% e 6%.

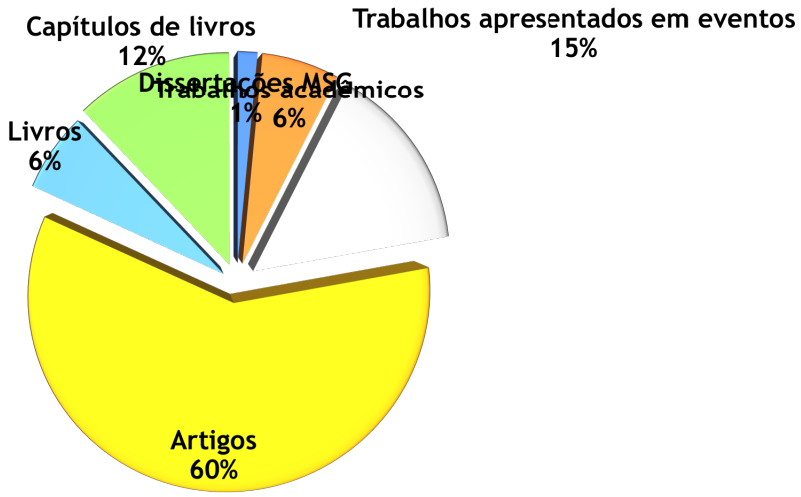


Gráfico 1: Distribuição do referencial teórico por tipo de publicação.

Fonte: Autor (2017), adaptado de Lima (2013)

Em termos de atualidade do referencial teórico, de acordo com o Gráfico 2, o mesmo apresenta 68% das referências publicadas há no máximo dez anos, uma vez que foi estabelecido tal critério como indicador na ocasião de realização da análise bibliométrica. Em contrapartida, 32% das referências estão fora do período estabelecido, onde a referência mais antiga data de 1969. Os materiais compreendidos nesses 32% são citados na pesquisa com a finalidade histórica, a fim de abordar as discussões iniciais sobre um determinado tema, ou ainda, porque os principais autores sobre o assunto não apresentam obras atualizadas, o que não justifica deixar de citá-los.

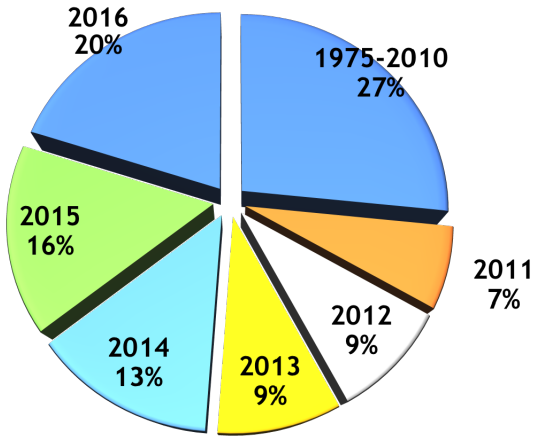


Gráfico 2: Distribuição do referencial teórico por ano de publicação.

Fonte: Autor (2017), adaptado de Lima (2013)

A figura 15 abaixo mostra um diagrama representando a distribuição da pesquisa:

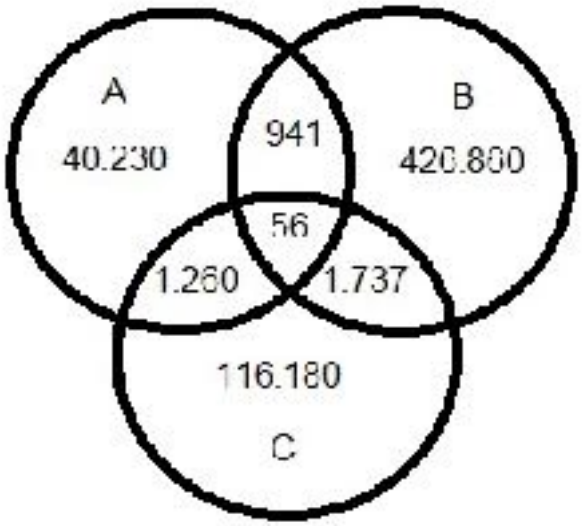


Figura 15 – Diagrama de Venn representando a pesquisa

Fonte: Autor (2017)

Dentre os 56 resultados obtidos no final da pesquisa, é válido destacar alguns, dentre eles:

ACES ground segment functionality and preliminary operational concept, que explica o Atomic Clock Ensemble no Espaço (ACES), que é uma missão da Agência Espacial Europeia (ESA) baseada em uma nova geração de relógios atômicos: um relógio de césio (PHARAO) baseado em átomos arrefecidos a laser e um ativo Space Hydrogen Maser (SHM). O Segmento Terrestre ACES (GS) é integrado dentro da arquitetura global do ISS, fornecendo as ligações de comunicação entre o solo e Espaço através do Columbus Control Center e NASA no segmento terrestre. A ACES GS é composta por dois componentes principais: o Centro de Suporte e Operações de Usuários (USOC) e uma rede de terminais terrestres MicroWaveLink (MWL GTs) controlados remotamente distribuídos em todo o mundo.O artigo apresenta a arquitetura de linha de base do Terreno ACES, suas principais funcionalidades e o conceito operacional preliminar.(DAGANZO et. al, 2009).

System overview of the Virginia Tech Ground Station, uma nova estacão terrestre colaborativa da Virginia Tech, foi feito um esforço colaborativo e multidisciplinar para construir uma nova estação terrestre via satélite na Virginia Tech, que é apoiada pela Faculdade de Engenharia da universidade e é capaz de suportar muitas missões de pesquisa, educacionais e amadoras. O projeto para a estação terrestre é fortemente baseado em software que fornece uma plataforma flexível permitindo a expansão futura. Cada subsistema possui uma fonte de software de rádio dedicada que pode ser emparelhada a um servidor quad-core através de um comutador de rede de alto rendimento; O switch oferece suporte a failover, permitindo que qualquer servidor de rádio acesse qualquer subsistema. A capacidade de transmissão e recepção é implementada principalmente usando a estrutura de rádio do software GNU Radio, enquanto a funcionalidade geral de comando e controle é implementada usando uma estrutura personalizada baseada no modelo de ator. O software de comando e controle fornece automação de sistemas, gerenciamento de usuários, programação e determinação de passagens, controle de hardware, registro e gerenciamento de dados para a estação terrestre. Os planos futuros incluem o suporte a uma plataforma de computação em nuvem que fornecerá redundância adicional e balanceamento de carga para a rede, além de fornecer uma estrutura personalizada de "Estação Terrestre como Serviço" (GSaS) que permite aos clientes utilizar a infraestrutura da nuvem para a missão de processamento dados. (HITEFIELD et. al, 2016).

Implementation of an actor framework for a ground station, artigo que trabalha em conjunto com o anterior, que descreve uma implementação para uma estrutura de estação terrestre baseada no modelo de ator e rádios definidos por software (SDRs), permitindo um sistema altamente escalável que pode operar em receptores de baixo custo ou escala para uma coleção de servidores. O modelo de ator define um sistema concorrente onde a unidade individual de computação é o ator e a comunicação entre componentes é tratada através de uma interface de passagem de mensagens. Pystation é uma estrutura de ator escrita em Python que implementa um esquema de passagem de mensagens onde cada ator tem a capacidade de se comunicar com atores ou atores locais em uma rede. Este artigo descreve algumas informações históricas e de fundo sobre o modelo de ator seguido por detalhes de sua implementação e exemplos de uso em uma estação terrestre. O GNU Radio é utilizado como o software para a camada física e o processamento de sinal, permitindo uma estação terrestre altamente robusta e flexível que pode ser alterada sem modificações tediosas de hardware. O

objetivo deste framework é fornecer a base de software para a Virginia Tech Ground Station (VTGS), a fim de aumentar a confiabilidade e flexibilidade. O projeto Virginia Tech Ground Station destina-se a servir de sandbox de operações terrestres e comunicações para atividades educacionais, como laboratórios de estudantes e projetos de pesquisa espacial.(DAVID et. al, 2016).

Estes trabalhos e artigos implementam e focam no mesmo ramo em que este trabalho detém seu objetivo, porém em nenhum artigo é proposto o que é neste. O objetivo do trabalho é desenvolver um módulo do software capaz auxiliar a operação, organização e distribuição das agendas de rastreamento de todas as estações terrestres da RIBRAS sem auxílio humano.