MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2024/07.26.12.14-TDI

ANÁLISE COMPARATIVA ENTRE OS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE DE OBDH DE NANOSATC-BR2 E AMAZONIA 1

Matilde de Jesus Silva

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologias Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelas Dras. Maria de Fátima Mattiello-Francisco, e Ana Maria Ambrósio, aprovada em 15 de maio de 2024.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/4BN9KU5>

INPE
São José dos Campos
2024

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2024/07.26.12.14-TDI

ANÁLISE COMPARATIVA ENTRE OS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE DE OBDH DE NANOSATC-BR2 E AMAZONIA 1

Matilde de Jesus Silva

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologias Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelas Dras. Maria de Fátima Mattiello-Francisco, e Ana Maria Ambrósio, aprovada em 15 de maio de 2024.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/4BN9KU5>

INPE
São José dos Campos
2024

Dados Internacionais de Catalogação na Publicação (CIP)

Silva, Matilde de Jesus.

Si38a Análise comparativa entre os processos de desenvolvimento de software de OBDH de NANOSATC-BR2 e Amazonia 1 / Matilde de Jesus Silva. – São José dos Campos : INPE, 2024.

xxiv + 105 p. ; (sid.inpe.br/mtc-m21d/2024/07.26.12.14-TDI)

Dissertação (Mestrado em Engenharia e Tecnologias Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2024.

Orientadoras : Dras. Maria de Fátima Mattiello-Francisco, e Ana Maria Ambrósio.

1. CubeSats. 2. Satélites de baixo custo. 3. Componentes COTS. 4. Computador de bordo (OBC). 5. Garantia de qualidade de software. I.Título.

CDU 629.78:004.4'2

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

DEFESA FINAL DE DISSERTAÇÃO DE MATILDE DE JESUS SILVA
BANCA Nº 087/2024, REG. 114960/2020

No dia 15 de maio de 2024, as 09:00h, por teleconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi APROVADO(A) pela Banca Examinadora, por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Engenharia e Tecnologias Espaciais / Engenharia e Gerenciamento de Sistemas Espaciais, com a exigência de que o trabalho final a ser publicado deverá incorporar as correções sugeridas pela Banca Examinadora, com revisão pelo(s) orientador(es).

Observações da banca:

A aprovação da banca foi unanime e condicionada a alterações do texto e a inclusão de um apêndice que traga as recomendações e as boas práticas para projetos do INPE. O novo documento gerado será referendado pela banca.

O não atendimento desses dois itens implicará na reprovação do trabalho da aluna.

Novo Título: "ANÁLISE COMPARATIVA ENTRE OS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE DE OBDH DE NANOSATC-BR2 E AMAZONIA 1"

Membros da Banca:

Dr. Maurício Gonçalves Vieira Ferreira – Presidente – INPE
Dra. Maria de Fátima Mattiello-Francisco – Orientadora – INPE
Dra. Ana Maria Ambrósio – Orientadora – INPE
Dr. Walter Abrahão dos Santos – Membro Interno – INPE
Dra. Regina Lúcia de Oliveira Moraes – Membro Externo – Universidade de Coimbra

Documento assinado eletronicamente por **Maria de Fatima Mattiello Francisco**, **Tecnologista**, em 29/07/2024, às 14:38 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Walter Abrahão dos Santos**, **Tecnologista**, em 29/07/2024, às 14:47 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mauricio Goncalves Vieira Ferreira**, **Coordenador de Rastreio, Controle e Recepção de Satélites**, em 29/07/2024, às 16:04 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ana Maria ambrosio (E)**, **Usuário Externo**, em 30/07/2024, às 14:10 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.mcti.gov.br/verifica.html, informando o código verificador **12127656** e o código CRC **259343A7**.

Referência: Processo nº 01340.003943/2024-81 SEI nº 12127656

"Mostra-me os teus caminhos, Senhor, ensina-me as tuas veredas; guia-me com a tua verdade e ensina-me, pois, tu és Deus, meu Salvador, e a minha esperança está em ti o tempo todo."

Salmos 25:4-5

Dedico esta dissertação ao meu amado pai, cuja memória sempre me inspirará a ser uma pessoa melhor. Sua presença, mesmo em ausência, é uma luz constante em minha vida.

À minha mãe, que, apesar das dificuldades e limitações impostas pela doença, sempre foi uma fonte de amor e coragem. Seu espírito de luta me dá forças para enfrentar os desafios do dia a dia.

E a Deus, por me sustentar e me guiar em todas as etapas desta jornada. Sem Sua graça e misericórdia, eu não teria chegado até aqui.

Esta conquista é para todos vocês!

AGRADECIMENTOS

A jornada desta dissertação não teria sido possível sem o apoio e a orientação de diversas pessoas especiais que estiveram ao meu lado. Gostaria de expressar meus sinceros agradecimentos a todos que, de alguma forma, contribuíram para este trabalho.

Primeiramente a Deus, por me sustentar e me guiar em todas as etapas desta jornada!

O meu profundo reconhecimento à Dra. Maria de Fátima Mattiello-Francisco, minha orientadora, por acreditar em mim desde o início, me guiar com paciência e fornecer orientação inestimável ao longo de todo o processo.

Minha gratidão à Dra. Ana Maria Ambrósio, coorientadora, pela atenção aos detalhes e pela paciência ao me ajudar a resolver os desafios que surgiram ao longo do caminho. Sua experiência e apoio foram essenciais para o desenvolvimento deste trabalho.

Ao Dr. Fabrício Kucinskis, membro interno suplente, minha mais profunda gratidão por não me deixar desistir. Sua amizade e encorajamento foram fundamentais para eu persistir nos momentos difíceis.

Expresso minha gratidão ao Dr. Walter Abrahão, membro interno da banca, pelo olhar crítico e sugestões pertinentes que enriqueceram o conteúdo desta dissertação.

Ao Dr. Maurício Ferreira, membro interno da banca, cuja experiência e sugestões valiosos ajudaram a moldar o rumo deste projeto.

Às Dras. Regina Lúcia de Oliveira Moraes e Eliane Martins, membros externos da banca, por aceitarem o convite a contribuírem com seu conhecimento para este trabalho.

Quero também agradecer a Dr. Sergio Mineiro, coordenador do Curso de Engenharia e Tecnologia Espaciais, e a toda a equipe de pós-graduação e biblioteca do INPE, pelo suporte e pelo empenho em proporcionar um ambiente propício para a pesquisa e a formação acadêmica.

Além do apoio acadêmico, também tive o privilégio de contar com a amizade e a espiritualidade da Missionária evangélica, Marta Malaquias, que me ajudou por meio de orações e aconselhamentos espirituais. Seu apoio foi uma fonte de força e inspiração.

A todos vocês, deixo meu mais sincero agradecimento. Sem o suporte de cada um, este trabalho não teria sido possível. Muito obrigada!

RESUMO

O advento dos Cubesats no início dos anos 2000 abriu novas oportunidades para pequenos satélites de baixo custo, através da padronização de subsistemas e do grande uso de componentes *Commercial of the Shelf* (COTS). Estudos de confiabilidade do Cubesat demonstram que o subsistema do computador de bordo (OBC) desempenha uma contribuição significativa nas falhas da missão. Os três principais subsistemas que causam falhas no CubeSat, entre 30 a 90 dias em órbita, são computadores de bordo (OBC) (16%), suprimento de energia (20%) e subsistema de comunicação (21%). Dado que o OBC é um subsistema intensivo em software, o esforço para garantia de qualidade segue as melhores práticas recomendadas pelos padrões de software espacial. Tais práticas são caras, demoradas e incompatíveis com o orçamento da maioria dos pequenos satélites. O presente trabalho de pesquisa tem por objetivo analisar os processos de desenvolvimento de software embarcado adotados pelo INPE em projetos de software de OBDH (*On-Board Data Handling*) embarcados em OBC de satélite de médio porte e de nanossatélite. Com foco nas melhores práticas recomendadas pela garantia de qualidade de software espacial, a análise está fundamentada nos processos de desenvolvimento de software embarcado no Amazonia-1, um satélite de médio porte, e no NanosatC-BR2, um nanossatélite baseado em Cubesat. Cada satélite foi desenvolvido em contexto diferente e por equipes com diferente nível de experiência. Apesar do tamanho do satélite, ambos os projetos enfrentaram situações semelhantes. O estudo foi motivado pelo desafio de garantir a qualidade do software embarcado em pequenas missões espaciais, considerando as restrições orçamentárias e as limitações de equipes reduzidas. Durante o desenvolvimento desses software, solicitações para alteração de requisitos são comuns, em geral decorrente da definição tardia do sistema, de atrasos na especificação de requisitos, da indisponibilidade de equipamentos e da falta de documentação em fases cruciais do desenvolvimento, entre outros. Para enfrentar esses desafios, a dissertação apresenta recomendações práticas para pequenas equipes que trabalham no desenvolvimento de software de bordo para satélites, enfatizando a necessidade de equilibrar qualidade e cronograma. A pesquisa sugere o uso de ferramentas de apoio a gestão de requisitos, a casos de teste e a rastreabilidade, ressaltando a importância de uma seleção cuidadosa, considerando o custo e os recursos disponíveis. A escolha da

ferramenta certa pode facilitar a gestão de mudanças e promover a colaboração entre as equipes, elementos essenciais para o sucesso do projeto.

Palavras-chave: CubeSats, satélites de baixo custo, componentes COTS, computador de bordo (OBC), garantia de qualidade de software, software espacial, OBDH (On-Board Data Handling), desenvolvimento de software embarcado, Amazonia-1, NanosatC-BR2, ferramentas de gestão de requisitos.

COMPARATIVE ANALYSIS OF OBDH SOFTWARE DEVELOPMENT PROCESSES FOR NANOSATC-BR2 AND AMAZONIA-1

ABSTRACT

The advent of CubeSats in the early 2000s opened new opportunities for low-cost small satellites through the standardization of subsystems and the extensive use of Commercial Off-The-Shelf (COTS) components. CubeSat reliability studies demonstrate that the onboard computer (OBC) subsystem significantly contributes to mission failures. The three main subsystems causing CubeSat failures, between 30 to 90 days in orbit, are onboard computers (OBC) (16%), power supply (20%), and communication subsystem (21%). Given that the OBC is a software-intensive subsystem, the effort for quality assurance follows the best practices recommended by space software standards. Such practices are expensive, time-consuming, and incompatible with the budget of most small satellites. This research aims to analyze the embedded software development processes adopted by INPE in OBDH (On-Board Data Handling) software projects embedded in OBCs of medium-sized satellites and nanosatellites. Focusing on the best practices recommended for space software quality assurance, the analysis is based on the development processes of software embedded in the Amazonia-1, a medium-sized satellite, and in the NanosatC-BR2, a nanosatellite based on CubeSat. Each satellite was developed in a different context and by teams with different levels of experience. Despite the satellite size, both projects faced similar situations. The study was motivated by the challenge of ensuring the quality of embedded software in small space missions, considering budget constraints and limited team sizes. During the development of these software projects, requests for requirement changes are common, generally due to late system definition, delays in requirement specification, unavailability of equipment, and lack of documentation in crucial development phases, among others. To address these challenges, the dissertation presents practical recommendations for small teams working on onboard software development for satellites, emphasizing the need to balance quality and schedule. The research suggests using tools to support requirements management, test cases, and traceability, highlighting the importance of careful selection considering

cost and available resources. Choosing the right tool can facilitate change management and promote collaboration among teams, essential elements for project success.

Keywords: CubeSats, low-cost satellites, COTS components, on-board computer (OBC), software quality assurance, space software, OBDH (On-Board Data Handling), embedded software development, Amazonia-1, NanosatC-BR2, requirements management tools.

LISTAS DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ACE	Attitude Control Electronics
AEB	Agência Espacial Brasileira
AIT	Assembly Integration Tests
AMSAT	The Radio Amateur Satellite Corporation
AOCS	Attitude and Orbit Control System
AR	Acceptance Review
AWDT	AWFI Data Transmitter
CDR	Detailed Design Review
cFS	Core Flight System
COESU	INPE/Coordenação Espacial do Sul
CoFI	Conformidade com Injeção de Falha
COM	Subsistema de Comunicação
COTS	Commercial of the Shelf
DDR	Digital Data Recorde
ECSS	European Cooperation for Space Standardization
EM	Engineer Model
EMSISTI	Space Systems & Technology
EPS	Subsistema de Energia Elétrica
ESA	Agência Espacial Europeia
E-ST	Space Engineering Standard – Engineering Standard
FATECs	Faculdades de Tecnologia do Estado de São Paulo
FEM	Fault Emulator Mechanism
FM	Flight Model
FRR	Flight Readiness Review
GPS	Global Positioning System
HIL	loop hardware in-the-loop
ICD	Software interface control document
INPE	Instituto Nacional de Pesquisas Espaciais
INVAP	Empresa Argentina de serviços de tecnologia

IOD	Initial Orbit Determination
IRD	Interface Requirements document
IRD	Software Interface Document
IRD	Software Interface Document
ITA	Instituto Tecnológico de Aeronáutica
LABRE	National association that represents Brazilian amateur radio before Brazilian and international authorities.
LACESM/ INPE	Laboratório de Ciências Espaciais
MBSE	Model-based systems engineering
MBT	Model-based Testing
MCTI	Ministério da Ciência, Tecnologia e Inovação
MDE	Model Driven Engineering
MDR	Mission Definition Review
MIL	model in-the-loop
M-ST	Space Project Management – Management Standard
NASA	Nacional da Aeronáutica e Espaço Americano
OBC	Computador de Bordo
OBDH	On Board Data Handling
OBSw	Softwere do OBC
ORR	Operation Readiness Review
PCDU	Power Conditioning and Distribution Unit
PDR	Preliminary Design Review
PMM	Plataforma MultiMissão
P-POD	Poly Picosatellite Orbital Deployer
PRR	Preliminary Requirements Review
QR	Qualification Review
QSEE	Qualidade do software Embarcado em Aplicaçõoes Espaciais
Q-ST	Space Product Assurance - Quality Standard
RB	Requirements Baseline
RTEMS	*Real-Time Executive for Multiprocessor Systems*
RTU	Remote Terminal Units

SADE	Solar Array Drive Electronic
SAMA	Anomalia Magnética do Atlântico Sul
SATS	Scalable Architecture Test System
SEI	Software Engineering Institute
SCF	Software configuration file
SDATF	Sistema de Determinação de Atitude Tolerante a Falhas
SDD	Software Design Document
SDP	Software Development Plan
SMDH	Santa Maria Design House
SPA	Space Project Assessment
SRD	Software Release Document
SReID	Software Released
SRevP	Software Review Plan
SRF	Software reuse file
SRR	System Requirements Review
SRS	Software Requirements Specification
SSS	Software System Specification
SUM	Software User Manual
SVaiP	Software Validation
SVerP	Software Verification Plan
SVR	Software verification report
SVR	Software Verification Report
SW	Software
SWE	Software Engineering
SWRR	Software Requirements Review
TC	Telecomando
TDR	Technical Delivery Review
TM	Telemetria
TRR	Technical Readiness Review
TS	Technical Specification
TT&C	Telemetry, Tracking and Command
UFABC	Universidade Federal do ABC

UFMG	Universidade Federal de Minas Gerais
UFRGS	Universidade Federal do Rio Grande do Sul
UFSM	Universidade Federal de Santa Maria
UHF	Ultra-high frequency
UML	Unified Modeling Language
V&V	Verificação e Validação
VHF	Very High Frequency
VSE	Very Small Entities
WFI	Wide-Field Imager
W.R. T	with Respect To

SUMÁRIO

1. INTRODUÇÃO

O número de missões espaciais baseadas em pequenos satélites, principalmente aqueles que adotaram o padrão Cubesat, vem crescendo nos últimos anos. Propostos em 1999 para utilização em projetos universitários, cujos principais objetivos eram puramente educacionais, esses satélites ganharam visibilidade e importância, principalmente pela padronização mecânica, elétrica e eletrônica de seus componentes, bem como pela adoção de componentes comerciais sem qualificação espacial, sob a argumentação de que riscos de perda de missão educacional são toleráveis. Essas medidas levam à redução de custos de missões e a uma maior variedade de equipamentos e soluções disponibilizados pela indústria. Sweeting (2018) destaca que, quanto mais equipamentos padronizados são usados em Cubesats, mais sua confiabilidade aumenta. Consequentemente, os Cubesats tornaram-se uma solução viável não só para missões com fins educativos, em que normalmente basta que o satélite seja lançado e opere durante alguns dias, mas também, para missões que vão além da função de qualificar novas tecnologias em órbita. Cubesats têm sido utilizados em aplicações de satélite nas quais os usuários dependem dos produtos da missão. O Sensoriamento Remoto está entre os principais setores a adotar o Cubesat, correspondendo a 45% das aplicações (VILLELA et al., 2019).

Apesar da crescente taxa de sucesso de missões com Cubesats em operação, a avaliação de confiabilidade apresentada em Langer e Boumeester (2016) considerando seis subsistemas de Cubesat, mostra que o Computador de Bordo (OBC) contribui significativamente para falhas de missão Cubesat. Segundo sua avaliação, os três subsistemas que mais causam falhas são Subsistema OBC, Subsistema de Suprimento de Energia (EPS) e Subsistema de Comunicação incluindo antenas (COM), correspondendo a 16%, 20% e 21%, respectivamente, das falhas de satélites ocorridas após ejeção, 30 e 90 dias em órbita.

O OBC é um subsistema intensivo de software, ou seja, suas funções são implementadas em sua maioria por software embarcado, portanto, as atividades de garantia de qualidade consistem no monitoramento e verificação das entregas estabelecidas no processo de desenvolvimento. As práticas de engenharia para o desenvolvimento de software espacial são estabelecidas em padrões espaciais como o ECSS-E-ST-40C (2009) e o NASA-STD-8739.8B (2022). Muitas empresas e instituições que fornecem software para satélites de

médio e grande porte adaptam os processos padrão de desenvolvimento de software a cada missão, o que é uma atividade cara e demorada, incompatível com orçamento de pequenas empresas. Diniz et al. (2019) estudaram fatores que impactam os processos de software e propuseram um método de adaptação de processos a fim de melhorar a qualidade do software produzido por equipes muito pequenas, do inglês, *Very Small Entities* (VSE).

As missões que utilizam o padrão Cubesat possuem, geralmente, orçamento limitado e, portanto, têm o desenvolvimento de software realizado por pequenas equipes que acumulam tarefas de garantia de qualidade no ciclo de projeto de software. A padronização do Cubesat beneficiou significativamente os processos de desenvolvimento de vários subsistemas com o uso de componentes COTS, porém não beneficiou igualmente o software embarcado. Pelo contrário, a miniaturização do hardware levou à inclusão de mais funções no software, como o software de cargas úteis que, em satélites maiores seriam realizadas por equipamentos dedicados, enquanto nos Cubesats são, normalmente, controlados pelo software do OBC.

A complexidade do software do OBC e, consequentemente, o seu custo, está mais relacionada com os requisitos que a missão deve cumprir do que com o tamanho do satélite. A adoção do padrão Cubesat na arquitetura da missão não garante que a complexidade do software OBC da missão será reduzida. Dependendo dos requisitos de uma missão que adota o padrão Cubesats, seu software a bordo pode ser tão ou mais complexo do que o de uma missão realizada por um satélite pesando centenas de quilogramas.

Embora existam estruturas de software incorporadas e bibliotecas disponíveis para utilização no setor espacial (uma pesquisa pode ser encontrada em Miranda et al. (2019)), o uso de código-fonte disponibilizado não garante a qualidade do produto final. A qualidade do software é fruto do seu processo de desenvolvimento e inclui as atividades de verificação e validação adotadas no processo, seja para nanosatélites, seja para satélites de médio porte.

Surgem então questões: como garantir a qualidade do software OBC de satélites com orçamento limitado e realizado por equipes muito pequenas? Como reduzir o processo de desenvolvimento de software OBC mantendo a qualidade de software exigida?

Essas questões representam um desafio significativo para equipes pequenas desenvolvedoras de software de bordo para missões fortemente limitadas em termos de orçamento.

O presente trabalho de pesquisa analisa experiências relatadas por diferentes equipes do Instituto Nacional de Pesquisas Espaciais (INPE), que contribuem na discussão dessas questões. São analisados os processos de software embarcado no OBC de dois diferentes satélites desenvolvidos: (i) o Amazônia1, um satélite de médio porte para Observação da Terra; e (ii) o NanosatC-BR2, um satélite baseado em Cubesat com fins acadêmicos e validação de novas tecnologias em órbita. As diferenças e semelhanças dos dois processos são destacadas do ponto de vista da garantia de qualidade, esforço de verificação e validação, artefatos de entrega, adesão aos padrões de qualidade da Cooperação Europeia para Normalização Espacial (ECSS) ECSS-Q-ST-80C (2009) e uso de ferramentas de apoio ao processo de desenvolvimento.

Este estudo visa contribuir para o entendimento dos desafios enfrentados no desenvolvimento de nanossatélites, fornecendo *insights* que possam orientar futuras pesquisas e iniciativas no campo emergente dos CubeSats.

1.1 Motivação

Desde 2012, o INPE/MCTI, intermediado pela Agência Espacial Brasileira (AEB), apoiam instituições acadêmicas brasileiras a conceber, testar e operar nanossatélites. Pioneiro no Brasil, o NanosatC-BR1, lançado em julho de 2014, faz parte do Programa NanosatC-BR, caracterizado pelo desenvolvimento de missões de cunho científico, tecnológico e educacional, com o uso de nanossatélites da classe padronizada conhecida como CubeSat. O Programa é essencialmente formado por alunos de graduação, bolsistas e professores da UFSM (Universidade Federal de Santa Maria, RS) que atuam em cooperação e parceria com pesquisadores, tecnologistas e alunos dos cursos de pós-graduação em Geofísica Espacial, e Engenharia e Tecnologias Espaciais do MCTI/INPE. O CubeSat AESP-14, lançado em 2015 em parceria com o Instituto Tecnológico da Aeronáutica (ITA) e o Instituto Nacional de Pesquisas Espaciais (INPE), intermediado pela Agência Espacial Brasileira (AEB), é outro marco importante nesse percurso, sendo seguido por outros projetos lançados, operacionais e em desenvolvimento desde então.

Entre estes destaca-se o NanosatC-BR2, cujo processo de desenvolvimento do OBDH foi alvo de estudo de Batista et al. (2020), que enfatizou que os nanossatélites baseados em CubeSats enfrentam desafios específicos. Esses satélites são compostos por componentes *Commercial Off-The-Shelf* (COTS) e seguem estruturas e interfaces padronizadas, o que permite a redução do ciclo de desenvolvimento de missões. O desafio reside na adaptação do processo de Verificação e Validação (V&V) para atender demandas voláteis de requisitos a serem implementados no software embarcado, mantendo a conformidade com padrões internacionais.

Outro desafio crítico é garantir a qualidade ao longo de todo o processo de desenvolvimento de software, minimizando a geração de artefatos, mas ainda assim atendendo à norma ECSS-E-ST-40C (2009). A necessidade de reduzir custos e cumprir ciclo de desenvolvimento curtos (cronogramas reduzidos em relação a satélite de maior porte) enquanto mantém padrões de segurança e eficiência requer uma abordagem inovadora no contexto dos nanossatélites (BATISTA et al., 2020).

Considerando a motivação do presente trabalho de pesquisa no propósito de agregar valor e fortalecer as práticas de garantia do produto adotadas em satélites de pequeno porte, como os CubeSats, destacam-se algumas contribuições científicas para melhoria do processo de desenvolvimento de software dos satélites no INPE como: adaptação às normas ECSS realizadas por fornecedores industriais de software (MATTIELLO-FRANCISCO, et al., 2007); abordagens de teste de software de bordo (AMBROSIO et al., 2007; PONTES et al, 2010); ferramentas automatizadas (CONCEIÇÃO, 2016), (CONCEIÇÃO, et al., 2019; BATISTA, et al., 2019); processos de teste (PONTES et al., 2014; PINHEIRO et al., 2014); estudos sobre o processo de desenvolvimento de software críticos para equipes pequenas (DINIZ, et al., 2019), entre outros. Em complemento às contribuições acadêmicas, adicionam-se relatos de contribuições práticas como a experiência vivenciada pela equipe de garantida do produto alocada para acompanhar o processo de desenvolvimento do software OBDH (*On Board Data Handling*) do satélite Amazonia-1. Nesse desenvolvimento foi utilizada uma ferramenta de gerenciamento de requisitos, na perspectiva de garantir a qualidade do software devido à sua eficácia no apoio ao processo, e por atender os requisitos da norma ECSS, cronogramas e custos do projeto de software.

A Engenharia de Requisitos destaca-se como uma área do conhecimento fundamental para garantir a qualidade no desenvolvimento e manutenção de sistemas de software. A prática eficaz dessa disciplina contribui significativamente para a redução de erros, falhas e ambiguidades nos produtos finais entregues (ECSS-Q-ST-80C, 2009) e é um dos pilares da garantia do produto.

Durante o desenvolvimento de software de bordo, solicitações para alteração de requisitos são comuns, em geral decorrente da definição tardia do sistema, de atrasos na especificação de requisitos de sistema e de interface, da indisponibilidade de equipamentos, da falta de documentação em fases cruciais do desenvolvimento, entre outros. Lidar com alterações de requisitos durante o desenvolvimento de software embarcado é uma questão afim ao processo de desenvolvimento de software adotado que, independentemente do modelo de desenvolvimento (incremental ou ágil) utilizado no processo, deve ser acompanhado por uma equipe que zela pela garantia da qualidade do software produzido.

A motivação da pesquisa dessa dissertação é identificar práticas da garantia da qualidade do software embarcado em satélites de médio porte que possam ser aplicadas em missões espaciais de Cubesats, considerando as restrições orçamentárias, ciclo curto de projeto e as limitações de equipes reduzidas em missões de nanossatélites.

1.2 Definição do problema

Aplicações espaciais dependem cada vez mais de sistemas computacionais para aumentar a autonomia de operações em satélite. Tais sistemas requerem software dedicados, sejam embarcados na eletrônica de voo dos veículos espaciais, sejam em solo contribuindo para a operação remota dos mesmos (AMBROSIO et al., 2005). A evolução constante no campo da tecnologia da informação impõe desafios significativos aos processos de desenvolvimento de software, demandando abordagens cada vez mais eficientes e assertivas.

No trabalho de Batista et al., (2020) foi destacado que as atividades de Verificação e Validação (V&V) são onerosas no processo desenvolvimento de sistemas intensivos em software (em inglês *Software-Intensive Systems* – SiS) aplicado aos nanossatélites, devido às constantes alterações dos requisitos e consequente necessidade de revalidação da

interoperabilidade na integração dos subsistemas. Devido a volatilidade dos requisitos e o maior risco aceitável em projeto de nanossatélites, o fechamento dos documentos de revisão é demorado, chegando a casos em que a liberação é feita em fases tardias no processo. Os problemas são geralmente identificados na fase de Montagem, Integração e Testes (em ingles, *Assembler, Integration and Testing* - AIT), quando inicia-se a integração do software com o satélite, primeiro na integração no modelo de Engenharia (em inglês, *Engineering Model* - EM), e muitas vezes extensível à integração no modelo Voo (em inglês, *Flight Model* - FM).

No estudo de caso do NanosatC-BR2, Batista et al. (2020) descreve os desafios enfrentados devido à natureza incremental e adaptativa dos requisitos durante o processo de desenvolvimento e à complexidade do software OBDH. Entretanto, sob o ponto de vista da garantia da qualidade, não aponta o uso de ferramentas de apoio à gestão e rastreabilidade dos requisitos. Já, no projeto do satélite de médio porte, com características semelhantes, e que contou com equipe dedicada à garantia da qualidade, o uso de uma ferramenta de gerenciamento de requisitos foi de grande valia para agilizar o processo, garantindo a aprovação e controle efetivo dos artefatos.

1.3 Objetivo

Esta dissertação analisa o processo de desenvolvimento de software para subsistemas OBC em satélites de diferentes portes, comparando as abordagens usadas no nanosatélite NanosatC-BR2 com as práticas adotadas no satélite de médio porte, o Amazônia-1, com objetivo de apresentar recomendações e melhores práticas para o desenvolvimento de software de bordo realizado por equipes pequenas e com restrições orçamentarias, que sejam viáveis em curto ciclo de projeto.

Objetiva-se trazer contribuições que orientem equipes pequenas no desenvolvimento de software para satélites, garantindo qualidade sem comprometer prazos de lançamento. O foco está na qualidade do software embarcado no computador de bordo, do inglês *On-Board Computer* (OBC), especialmente considerando as restrições de recursos e as equipes reduzidas envolvidas nos projetos de pequenos satélites que adotam o padrão CubeSat.

A análise busca identificar desafios comuns, como requisitos instáveis, atrasos na entrega de equipamentos e falta de documentação adequada. Além disso, visa avaliar o equilíbrio entre a qualidade do software, em conformidade aos padrões internacionais como a ECSS-E-ST-40C (2009) e os prazos de entrega reduzidos, comuns em missões que adotam o padrão Cubesat. O uso de boas práticas, tais como ferramentas de suporte à gestão de requisitos e de rastreabilidade, e a adoção de abordagens adequadas à integração e testes para validação dos requisitos são também analisados na perspectiva de viabilizar a entrega dentro dos prazos estipulados.

Outro objetivo é trazer ao conhecimento dois cenários vivenciados no INPE no processo de desenvolvimento de software espacial, com base nas dificuldades identificadas nos projetos Amazonia-1 e NanosatC-BR2. Os projetos avaliados são de diferentes portes, porém apresentaram dificuldades semelhantes no processo de desenvolvimento de software embarcado.

1.4 Estrutura da Dissertação

O restante dessa dissertação está organizado da seguinte forma: Capítulo 2 - Fundamentos Teóricos; Capítulo 3 - Metodologia; No Capítulo 4, Trabalhos Relacionados, são apresentados trabalhos do INPE que contribuem para o processo de desenvolvimento do software e trabalhos externos que contribuem com lições aprendidas e soluções para a melhoria do processo; o Capítulo 5, Software Embarcado OBDH – Amazonia-1, descreve o processo de desenvolvimento do OBDH do Amazonia-1. O Capítulo 6, Software Embarcado do NanosatC-BR2, descreve o processo de desenvolvimento do OBDH do NanosatC-Br2, referido nesta dissertação por OBSw para facilitar sua distinção do OBDH do Amazonia-1; no Capítulo 7, Análise Comparativa do Processo de Desenvolvimento do Software dos Satélites NanosatC-BR2 e Amazonia-1, são discutidas similaridades e especificidades dos dois processos apresentados, à luz das práticas preconizadas pelas normas ECSS, da garantia da qualidade do software embarcado. O Capítulo 8 conclui o trabalho e apresenta propostas para pesquisa futura.

O Apêndice A traz um guia de uso da ferramenta JAMA. O Apêndice B apresenta o resultado do estudo de algumas ferramentas para Gestão e rastreabilidade de requisitos. O anexo A apresenta uma tabela com as contribuições do JAMA para o software do satélite Amazonia-1 elaborado pela equipe de desenvolvimento do Grupo de Supervisão

de Bordo do INPE.

2. FUNDAMENTOS TEÓRICOS

2.1 Normas ECSS

A Cooperação Europeia para a Normalização Espacial (do inglês, *European Cooperation for Space Standardization* - ECSS) é uma iniciativa da Agência Espacial Europeia que produziu um conjunto de normas para orientar a gestão, engenharia e garantia de qualidade de produtos espaciais desde 1993 (KRIEDTE, 1995). As normas ECSS têm sido usadas em todo o mundo, ajudando a melhorar agências e associações industriais no setor espacial.

As Normas ECSS abrangem três categorias distintas: Gerenciamento (-M-ST), Engenharia (-E-ST) e Qualidade (-Q-ST) cada uma desempenhando um papel essencial nas práticas de gerenciamento, engenharia e garantia de produtos em projetos espaciais. Esta série estabelece diretrizes harmonizadas para o desenvolvimento e execução de projetos e aplicações espaciais. (ECSS-Q-ST-80C, 2009) (ECSS-E-ST-40C, 2009).

O propósito central das Normas ECSS é definir padrões comuns que transcendam as fronteiras organizacionais, permitindo a interoperabilidade entre entidades diversas envolvidas em projetos espaciais. A abordagem adotada por essas normas se diferencia por se concentrar no "o quê" deve ser realizado, em vez de prescrever detalhadamente "como" organizar e executar as tarefas específicas. Tal enfoque estratégico oferece flexibilidade para a aplicação de estruturas e métodos organizacionais já existentes, sempre que estes demonstrarem eficácia.

A norma ECSS-Q-ST-80C faz interface com engenharia e gerenciamento espacial, que são abordados nos ramos de Engenharia (-E) e Gestão (-M) da ECSS Sistema e explica como eles se relacionam com a garantia do produto de software.

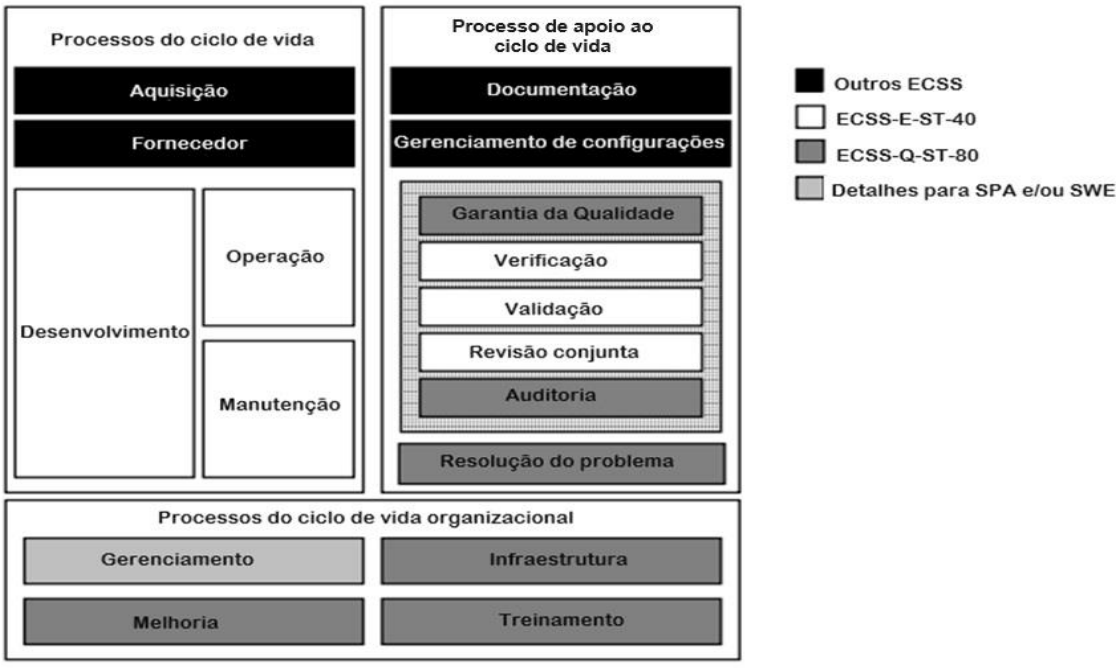
A garantia da qualidade do produto de software desempenha papel fundamental no desenvolvimento de software. É a disciplina responsável por acompanhar o processo de desenvolvimento de software adotado, com o propósito de evitar erros e problemas que podem comprometer a qualidade do produto final.

O gerenciamento da qualidade de software preocupa-se em garantir que os sistemas de software sejam executados de modo eficiente e confiável e sejam entregues no prazo e dentro do orçamento. O uso de técnicas de gerenciamento da qualidade, somado às

tecnologias de software e métodos de teste, levou a melhorias significativas no nível da qualidade de software ao longo dos últimos 20 anos (SOMMERVILLE, 2014).

Em conformidade com a ECSS, os processos relacionados a software são agrupados em 3 blocos, apresentados na Figura 2.1: (i) o grupo dos processos do ciclo de vida do software, onde se encontram o processo de desenvolvimento, o processo de operação e o processo de manutenção do software ; e (ii) o grupo dos processos de suporte ao ciclo de vida, onde se encontram os processos de garantia da qualidade, de Verificação, de Validação, de Revisões conjuntas e de Auditoria; e (iii) o grupo dos processos organizacionais do ciclo de vida. Os processos de interesse desta dissertação estão nos grupos (i), observados na coluna esquerda da Figura 2.1, e o grupo (ii), encontrados na coluna direita da Figura 2.1.

Figura 2.1 - Processos relacionados a software no padrão ECSS.



Fonte: ECSS-E-ST-40C (2019).

A seguir apresenta-se um detalhamento dos processos associados aos grupos (i) e (ii) da Figura 2.1 e associações com normas específicas, como ECSS-E-ST-40C (para engenharia de software) e ECSS-Q-ST-80C (para garantia de qualidade de software), objetos de estudo do presente trabalho de pesquisa.

(i) Processos do Ciclo de Vida

Aquisição: Envolve os processos de obtenção de software, seja através de desenvolvimento interno ou aquisição de fornecedores externos. Norma Associada: Outros ECSS (indicado em preto).

Fornecedor: Relaciona-se com a gestão de fornecedores que fornecem componentes de software ou serviços. Norma Associada: Outros ECSS (indicado em preto).

Desenvolvimento: Abrange todas as atividades envolvidas no desenvolvimento de software, desde a especificação de requisitos até o design, implementação, testes e integração. Norma Associada: ECSS-E-ST-40 (indicado em branco).

Operação: Inclui as atividades necessárias para operar o software uma vez que ele está em uso, garantindo que ele funcione conforme esperado. Norma Associada: ECSS-E-ST-40 (indicado em branco).

Manutenção: Envolve a correção de defeitos, atualizações e modificações necessárias para adaptar o software a novos requisitos ou ambientes. Norma Associada: ECSS-E-ST-40 (indicado em branco).

(ii) Processos de Apoio ao Ciclo de Vida

Documentação: Envolve a criação e manutenção de toda a documentação necessária para suportar o desenvolvimento e manutenção do software. Norma Associada: Outros ECSS (indicado em preto).

Gerenciamento de Configurações: Abrange o controle de mudanças e a manutenção da integridade e rastreabilidade dos itens de configuração do software. Norma Associada: Outros ECSS (indicado em preto).

Garantia da Qualidade: Inclui atividades de garantia de que o software atende aos requisitos de qualidade especificados. Norma Associada: ECSS-Q-ST-80C (indicado em cinza).

Verificação: Processo de assegurar que os produtos intermediários e finais atendem às especificações e requisitos estabelecidos. Norma Associada: ECSS-Q-ST-80C (indicado em cinza).

Validação: Processo de garantir que o software atende às necessidades e expectativas dos usuários finais. Norma Associada: ECSS-Q-ST-80C (indicado em cinza).

Revisão Conjunta: Atividades de revisão realizadas conjuntamente pelas partes interessadas para assegurar que o desenvolvimento está no caminho certo. Norma Associada: ECSS-Q-ST-80C (indicado em cinza).

Auditoria: Revisões formais para assegurar a conformidade com os processos e normas estabelecidas. Norma Associada: ECSS-Q-ST-80C (indicado em cinza).

Resolução do Problema: Processos para identificar, analisar e resolver problemas e defeitos encontrados no software. Norma Associada: Outros ECSS (indicado em preto).

Dentre os processos descritos acima, destacam-se em negrito como relevantes para o presente trabalho: o **processo de desenvolvimento do software** associado ao grupo (i) processos do ciclo de vida, e os **processos de verificação, de validação e de revisão conjunta de garantia da qualidade e de auditoria,** associados ao grupo (ii) processos de apoio ao ciclo de vida).

O processo de desenvolvimento do software, em geral é estabelecido pelo fornecedor do software e adota o modelo de desenvolvimento praticado pela equipe treinada pelo fornecedor.

O processo de Verificação é em geral estabelecido pelo cliente e sua condução é apoiada pela equipe de garantia da qualidade. Várias técnicas de verificação podem ser adotadas, tais como revisão, inspeção, teste, acompanhamento, leitura cruzada, verificação documental, modelo de simulação e muitos tipos de análise, tais como análise de rastreabilidade, prova formal ou árvore de falhas análise. (ECSS-Q-ST-80C, 2009).

Teste é uma técnica de verificação altamente recomendada no desenvolvimento de software porque permite executar dinamicamente o código fonte, em ambiente operacional ou simulado, a partir de entradas especificadas e verificar se a saída está conforme esperado.

Os testes devem ser realizados de acordo com uma estratégia para cada nível de teste (ou seja, unidade, integração, validação em relação à Especificação Técnica, validação em relação aos Requisitos de Base, aceitação), que inclui (ECSS-Q-ST-80C, 2009):

1. os tipos de testes a serem realizados; por exemplo: funcional, limites, desempenho, usabilidade, entre outros;
2. os testes a serem realizados de acordo com os planos e procedimentos;
3. os meios e organizações para desempenhar a função de garantia de teste e validação.

Em 2016, devido ao aumento da demanda de missão de nanossatélites para fins de demonstração de novas tecnologias em órbita, com uso do padrão Cubesata Agência Espacial Europeia (ESA) divulgou o documento intitulado "*ECSS Engineering Standard Tailored for In-Orbit Demonstration (IOD) CubeSat Projects*" (ESA, 2024). Os IOD Cubesats são considerados estruturas (1-U ou múltiplas) geralmente caracterizadas pelos seguintes atributos: (i) Sistemas autônomos completos incluindo plataforma, carga útil, segmento terrestre e operações; (ii) Maior perfil de aceitação de risco; (iii) Baixo nível de complexidade (relativo a outro projeto espacial da ESA); (iv) Baixo custo e curto prazo; (v) Ciclo de vida operacional curto (menos de um ano); (vi) Aceitação de ponto único de falhas; (vii) Redundância limitada e tolerância a falhas; (viii) Modo de segurança robusto; (ix) Uso extensivo de elementos COTS e testes extensivos com foco no nível do sistema (funcional e ambiente, qualificação e aceitação); (x) Organização simples do projeto com equipes bem integradas, poucos fornecedores ou subcontratados. Uma descrição desses atributos é dada na Tabela 2.1.

Tabela 2.1 - Atributos IOD - CubeSats conforme ECSS.

Atributo	Descrição
Sistemas Autônomos Completos	Inclui a plataforma, carga útil, segmento terrestre e operações, assegurando que o CubeSat funcione de forma independente.
Maior Perfil de Aceitação de Risco	Aceitação de níveis mais altos de risco em comparação com missões espaciais tradicionais, devido à natureza experimental e de demonstração dos CubeSats.
Baixo Nível de Complexidade	Comparativamente menos complexo que outros projetos espaciais da ESA, facilitando o desenvolvimento e implementação rápidos.
Baixo Custo	Projetos são desenvolvidos com orçamento limitado.
Curto Prazo	Cronogramas curtos, otimizando recursos e tempo de desenvolvimento.
Ciclo de Vida Operacional Curto	Tempo de operação típico é inferior a um ano, permitindo rápida iteração e validação de tecnologias.
Aceitação de Ponto Único de Falha	Reconhecimento de que a falha de um único componente pode levar à falha da missão, aceitando este risco como parte do perfil operacional.
Redundância Limitada e Tolerância a Falhas	Sistemas são projetados com redundância mínima, sendo a tolerância a falhas um aspecto menos prioritário.
Modo de Segurança Robusto	Implementação de modos de segurança para garantir que o CubeSat possa entrar em um estado seguro em caso de falha ou anomalia.
Uso Extensivo de Elementos COTS	Utilização intensiva de componentes comerciais prontos para uso (COTS), que são mais econômicos e rápidos de integrar.
Testes Extensivos no Nível do Sistema	Realização de testes funcionais e ambientais completos, incluindo qualificação e aceitação, para assegurar o desempenho esperado.
Organização Simples do Projeto	Estrutura de projeto simplificada com equipes integradas e poucos fornecedores ou subcontratados, promovendo eficiência na comunicação e execução.

Fonte: IOD Cubesat Project (2024).

2.2 Fundamentos do processo de desenvolvimento de software e garantia da qualidade preconizada nas normas espaciais

De acordo com a hierarquia na árvore de produto ECSS, o software OBDH é um componente do subsistema OBC e, por essa razão seus requisitos são especificados pelo subsistema OBC que, por sua vez tem seus requisitos definidos em nível de sistema

(satélite). Assim, a entrega do código fonte do software desenvolvimento deve ser validada no mínimo em 3 níveis: embarcado no hardware OBC, integrado ao subsistema OBC, e integrado ao satélite (sistema). Desta forma o processo de desenvolvimento do software e o processo de garantia da qualidade associado devem estar enquadrados no ciclo nas fases do ciclo de vida do projeto espacial.

O ciclo de vida de um projeto espacial segue uma abordagem bem definida, devido à natureza crítica das missões e à necessidade de garantir alta confiabilidade e qualidade.

Uma visão mais detalhada do ciclo de vida de projetos espaciais, suas fases e respectivas revisões conforme a norma ECSS-Q-ST-80C (2009) é dada a seguir:

Fase 0: Desenvolvimento da declaração de missão e identificação preliminar de conceitos de missão. Encerra-se com a revisão da definição de missão (MDR);

Fase A: Proposição e desenvolvimento de estudos de conceitos de missão e avaliação de arquiteturas. Encerra-se com a revisão preliminar de requisitos (PRR);

Fase B: Definição das especificações de sistema e projeto preliminar. Encerra-se com a revisão de projeto preliminar (SRR Software) (PDR);

Fase C: Fabricação do modelo de engenharia e desenvolvimento do projeto detalhado do sistema. Encerra-se com a revisão de projeto detalhado (CDR)

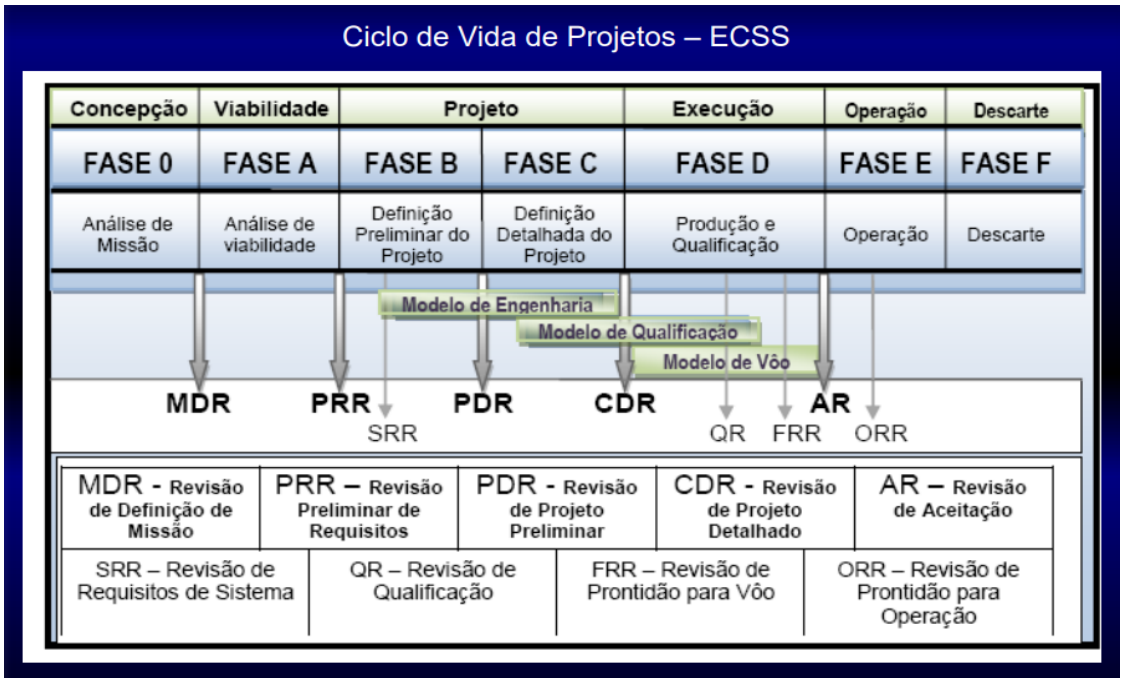
Fase D: Fabricação do modelo de qualificação e do modelo de voo, garantindo que o produto atenda aos requisitos do projeto. Encerra-se com a revisão de qualificação (QR) e a revisão de aceitação (AR);

Fase E: Operação do satélite;

Fase F: Descarte do satélite.

A Figura 2.2 apresenta as fases e revisões do ciclo de vida de projetos espaciais. (ALBUQUERQUE e PERONDI, 2010) Destacam-se na figura as barras: (i) Modelos de Engenharia, (ii) Modelo de Qualificação e (iii) Modelo de Voo, cujas atividades de desenvolvimento, fabricação, integração e testes ocorrem nas Fases B, C e D.

Figura 2.2 - Principais marcos dos processos ao longo das fases do ciclo de vida de um projeto.



Fonte: Albuquerque e Perondi (2010).

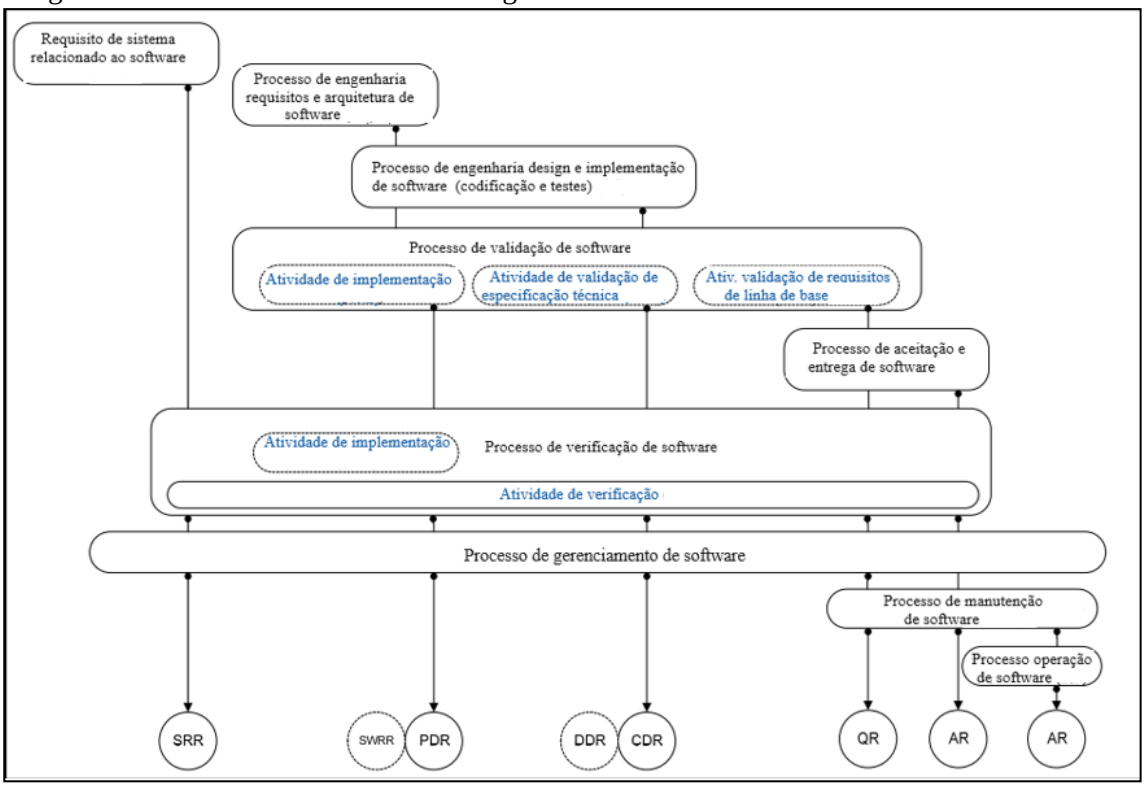
No início da fase B, um documento Plano de Desenvolvimento de Software define e instancia em detalhes o processo de desenvolvimento do software, em conformidade com as normas (ECSS-Q-ST-80C e ECSS-E-ST-40C), as fases, revisões e entregas do projeto de software. A Figura 2.3 identifica os principais processos simultâneos de engenharia de software e mostra como e quando eles são sincronizados pelas avaliações do cliente/fornecedor.

No processo de desenvolvimento do software (também referido por processo de engenharia do software), avaliações são feitas em revisões, nomeadamente, SRR (Revisão de Requisitos do Subsistema onde o componente software será embarcado), SWRR (Revisão de Requisitos de Software), PDR (Revisão de Projeto Preliminar), DDR (Revisão de Projeto Detalhado do Software), CDR (Revisão Crítica de Projeto), QR (Revisão de Qualificação), AR (Revisão de Aceitação). As revisões são essenciais para garantir a conformidade do sistema, dos subsistemas e componentes com os requisitos estabelecidos em cada nível.

Independentemente do porte do satélite, o desenvolvimento do software embarcado em um subsistema do satélite ocorre nas fases B e C, quando os testes de validação dos

requisitos de sistema são realizados no Modelo de Engenharia. Retrabalhos no código fonte são previstos na fase D, somente em caso excepcionais. Assim, a entrega do software é esperada antes da revisão CDR do satélite que, em via de regra, ocorre no final da fase C.

Figura 2.3 - Processos simultâneos de engenharia de software no Ciclo de Vida do software.



Fonte: ECSS-E-ST-40C (2009).

Independentemente do porte do satélite, o desenvolvimento do software embarcado em um subsistema do satélite ocorre nas fases B e C, quando os testes de validação dos requisitos de sistema são realizados no Modelo de Engenharia. Retrabalhos no código fonte são previstos na fase D, somente em caso excepcionais. Assim, a entrega do software é esperada antes da revisão CDR do satélite que, em via de regra, ocorre no final da fase C.

A norma ECSS-Q-ST-80C (*Software Product assurance*) Garantia da Qualidade ECSS define os Pilares Fundamentais no Desenvolvimento de Software levando em consideração a família ISO 9000 existente de documentos e a norma ISO/IEC 12207. (ECSS-Q-ST-80C, 2009). Uma equipe de Garantia de Qualidade é responsável por

verificar a conclusão das atividades de verificação e validação de software do subsistema OBC ao longo de todo o processo de desenvolvimento, com base no Plano de Desenvolvimento de Software (PDS). A definição dos artefatos do projeto de software, como documentos e código-fonte do software, bem como as ferramentas a serem utilizadas no gerenciamento de requisitos e testes de software, fazem parte do PDS e são fundamentais para apoiar a equipe de Garantia da Qualidade na execução de suas atividades. Igualmente relevante para a garantia da qualidade é a definição de um Plano de Verificação.

A verificação inclui técnicas, tais como, revisão, inspeção, testes, modelagem, simulação e muitos tipos de análise, como rastreabilidade, prova formal, árvore de falhas (ECSS-Q-ST-80C, 2009). Os testes devem ser realizados de acordo com uma estratégia para cada nível de detalhe (unidade, integração, validação da especificação técnica, dos requisitos, aceitação, etc.). Planos e procedimentos devem ser escritos para todos os tipos de testes.

A Tabela 2.2 descreve as fases do ciclo de vida de um projeto espacial segundo a norma ECSS-M-ST-10C e relaciona essas fases aos processos do ciclo de vida do software conforme especificado no documento ECSS-E-ST-40C. A células marcadas com XX destacam os relacionamentos considerados possíveis, porém não desejáveis.

Tabela 2.2 – Ciclo de Vida o Desenvolvimento do Software Conforme ECSS.

Ciclo de Vida ECSS ECSS-M-ST-10C =>	**Fase B Definição Preliminar do Projeto (SRR)**	**Fase B Definição Preliminar do Projeto (PDR)**	**Fase C Definição Detalhada do Projeto (CDR)**	**Fase D Produção e Qualificação (QR)**	**Fase D Produção e Aceitação (AR)**
Requisito de sistema relacionado ao software	X				
Processo de gerenciamento de software	X	X	X	XX	XX
Processo de engenharia de requisitos e arquitetura de software		X			
Processo de engenharia design e implementação de software (codificação e testes)			X		
Processo de validação de software - Atividade de implementação, de validação de especificação técnica e de validação de requisitos de base		X	X	XX	XX
Processo de aceitação e entrega de software			X	XX	XX
Processo de verificação de software - Atividade de implementação e de verificação	X	X	X	XX	XX
Processo de operação de software					X
Processo de manutenção de software				X	X

2.3 Simuladores na validação de software: uma abordagem abrangente

De forma genérica, no contexto de projeto e operação de satélites, a categoria de simulador abrange uma extensa gama de atividades relacionadas à modelagem e simulação, adotadas nas diferentes fases do ciclo de vida de um satélite. Em consonância com as diretrizes preconizadas pela Agência Espacial Europeia (ECSS-E-TM-10-21A, 2010), esta seção explorará a natureza e a relevância dos simuladores no âmbito da validação do software em projetos espaciais. Os simuladores, neste contexto, consistem em modelos de simulação e infraestrutura que desempenham um papel fundamental em diversas fases do ciclo de vida de um programa espacial. Estas fases abarcam desde a fase inicial de especificação até as operações em curso do sistema. A utilidade dos simuladores transcende a mera imitação de condições operacionais; eles são peças-chave no suporte a atividades críticas de engenharia e operacionais.

A aplicação da simulação abrange desde a validação de requisitos e especificações até a verificação do projeto e, por fim, facilitando operações eficazes. O emprego de

simuladores nesse contexto não se limita a uma única fase do ciclo de vida do programa; ao contrário, é considerado um elemento-chave em cada etapa, contribuindo significativamente para o sucesso global do empreendimento (EICKHOFF, 2009).

É crucial ressaltar que, conforme as diretrizes delineadas pela ECSS, a simulação deve ser integrada de forma intrínseca ao processo de engenharia do software. A inserção precoce e contínua dos simuladores nesse processo é essencial para extrair o máximo benefício de seu uso, incluindo o apoio na verificação de requisitos de software nas fases iniciais do projeto do satélite. Esta integração não apenas valida o software em condições representativas, mas também oferece *insights* valiosos para aprimorar o projeto e otimizar a operacionalidade do sistema espacial.

Dado que o software OBDH é um "componente" do subsistema OBC, o seu desenvolvimento ocorre em paralelo com o desenvolvimento de vários outros subsistemas do satélite. O uso de simuladores torna-se essencial na validação do software, pois nem todos os elementos reais de outros subsistemas com os quais o software deve interagir estarão disponíveis no momento da validação do software.

2.4 Ferramentas de apoio à gestão e qualidade no desenvolvimento de software: Uma perspectiva abrangente

As ferramentas de gestão destinadas a respaldar o processo de desenvolvimento de software desempenham um papel muito importante na eficiência e eficácia das operações, principalmente quando se trata de gestão e garantia da qualidade. (BARTIÉ, 2002).

Primordialmente, essas ferramentas são projetadas para orientar e aprimorar o planejamento das atividades de teste. Elas desempenham um papel fundamental na definição de escopos, na escolha de abordagens apropriadas, na alocação de recursos adequados e na programação eficiente de reuniões e atividades. Ao agir como catalisadores do processo de planejamento, essas ferramentas contribuem para a organização estruturada e eficaz das tarefas relacionadas ao desenvolvimento de software.

Além do suporte ao planejamento, essas ferramentas desempenham um papel vital na coleta inicial de requisitos, facilitando e agilizando essa fase do desenvolvimento. Elas oferecem assistência no processo de documentação, rastreando modificações e controlando as versões do software. Ao gerenciar esse aspecto, as ferramentas contribuem

para a criação e manutenção de padronizações, promovendo a consistência e a qualidade ao longo do processo.

Uma característica distintiva dessas ferramentas é sua capacidade de auxiliar na elaboração de estimativas de tempo e custos. Ao proporcionar *insights* baseados em dados reais do desenvolvimento, essas ferramentas capacitam os gestores a tomarem decisões com bases em informações, contribuindo para um planejamento mais preciso e eficiente. Adicionalmente, as ferramentas têm a capacidade de dimensionar as equipes de acordo com o tempo disponível, otimizando a alocação de recursos humanos.

Nesse contexto, as considerações de Bartié (2002) ressaltam a importância estratégica dessas ferramentas como elementos integrados ao processo de desenvolvimento de software. Sua contribuição, que vai desde o planejamento e documentação até a estimativa de tempo e custos, destaca essas ferramentas como "aliadas essenciais" na busca pela qualidade e eficiência no cenário complexo do desenvolvimento de software. Essa perspectiva reforça a necessidade de uma abordagem integrada e bem-sucedida no uso dessas ferramentas, consolidando seu papel como pilares fundamentais na gestão e garantia da qualidade do processo de desenvolvimento de software.

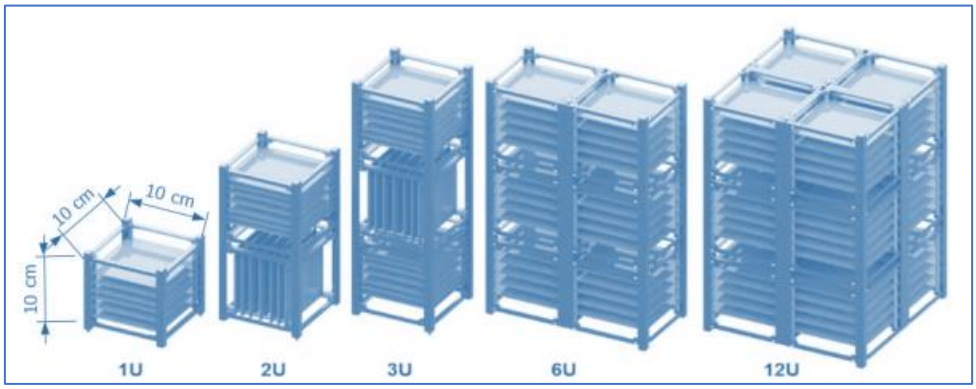
2.5 Principais características físicas dos satélites

As principais características do NanosatC-BR2 estão dentro das categorias dos cubesats, conforme seguem:

- Satélites compostos por unidades padronizadas cúbicas de 1U (100 x 100 x 100 mm), formando composições de 2U, 3U, ou mais.
- Uso de sistemas de ejeção em órbita, denominados de P-POD (Poly Picosatellite Orbital Deployer), capazes de liberar diversos satélites pela mesma interface. Existem P-POD (sistema de implantação padrão que garante que todos os desenvolvedores do CubeSat estejam em conformidade com os requisitos físicos comuns) comerciais destinados a satélites 1U, 2U, 3U e 6U.
- Uso de componentes eletrônicos comerciais (COTS) nos subsistemas de bordo, não necessariamente qualificados para uso espacial. (MCTI, 2017).

Na Figura 2.4 são mostradas as dimensões de um cubesat e suas configurações usuais.

Figura 2.4 - Algumas configurações de cubesats.



Fonte: MCTI (2017).

Nanossatélites como exemplo NanosatC-BR2, fazem parte da categoria dos Cubesats dentro da classe conhecida como satélites de pequeno porte, com massa entre 1 e 10 kg.

Na Tabela 2.3 é mostrada, para efeito de comparação, a classificação de alguns satélites conforme sua massa.

Tabela 2.3 - Algumas categorias de satélites de acordo com sua massa.

Categoria	Classe	Massa faixa (kg)
Padrão Satelites	Grande	>1000
		500-1,000
Pequeno Satelites	Mini	100-500
	Micro	10-100
	Nano	1.0-10
	Pico	0.1-1.0
	Femto	<0.1

Fonte: Rohling (2018).

Os satélites da série Amazônia são formados por dois módulos independentes: um Módulo de Serviço, chamado Plataforma Multimissão (PMM), e um Módulo de Carga Útil, que abriga câmeras imageadoras e equipamentos de gravação e transmissão de dados de imagens e os painéis solares. Os sistemas satelitais como Amazonia-1, são artefatos que possuem uma estrutura de médio porte por suportar a capacidade de carga total de 2 mil a 1800 kg a altitudes entre 600 e 900 km e com alta complexidade em

desenvolvimento (WAYRONE et al., 2022).

3. METODOLOGIA

Este trabalho se iniciou pelo estudo do processo de desenvolvimento do software de OBDH do Amazonia-1, a partir da vasta (e restrita) documentação de projeto, dos relatos dos participantes e da experiência dos autores no processo, nos papéis de desenvolvedores e de garantia de produto.

O satélite Amazonia-1 é um projeto institucional liderado pelo INPE com participação de múltiplas empresas nacionais e internacionais na qualidade de fornecedores. Por essa razão detalhes das implementações do software, regras de operação e de tratamento de falhas, não constam dos relatos apresentados no presente trabalho. Constam apenas as informações suficientes para a descrição do processo, dos desafios encontrados e dos resultados finais. As etapas do processo e os produtos gerados foram elencados e comparados com aqueles previstos pelas normas da ECSS, notadamente do ramo de qualidade de software. Uma análise qualitativa foi realizada sobre os documentos "faltantes" no caso do Amazonia-1, e seu impacto sobre o desenvolvimento, sempre sob a óptica de custo versus benefício, em um projeto com equipe reduzida, cronograma restrito e recursos limitados. Em seguida foi estudado o processo de desenvolvimento de um satélite de pequeno porte, o NanosatC-BR2 visando identificar suas práticas e processos que levaram ao sucesso dessa missão.

A disparidade nas dimensões dos satélites sugere diferenças significativas nos processos de desenvolvimento. Gonçalves et al. (2017) evidencia que uma das principais dificuldades encontradas nos projetos de satélites está na carência de um processo estruturado de gerenciamento de projeto. Busca-se analisar diferenças e similaridades, reconhecer lições aprendidas e recomendar boas práticas a serem adotadas nos processos de desenvolvimento de software, bem como nos testes, focando-se, especialmente, na implementação eficaz do gerenciamento e rastreabilidade de requisitos, bem como verificação e validação do sofware alvo de desenvolvimento.

Com objetivo de identificar as principais dificuldades enfrentadas nesses projetos, bem como similaridades e diferenças nos processos de desenvolvimento do software, de garantia da qualidade, de verificação e de validação, descritos na Seção 2.1, elabora-se uma análise comparativa entre as características de desenvolvimento de software OBDH embarcado em um Cubesat e um satélite de médio porte, especificamente nos NanosatC-

BR2 e Amazonia-1, respectivamente.

Identificando pontos em comum no processo de desenvolvimento do software OBDH dos satélites Amazonia-1 e NanosatC-BR2, o estudo é fundamentado nas definições da ECSS/IOD que caracterizam as missões Cubesat. Com base nos atributos de uma missão Cubesat apresentadas pelo IOD pretende-se identificar boas práticas de garantia da qualidade do software que possam ser aplicadas a processo de desenovlivmento e processos de verificação e validação de software de Cubesats.

4. TRABALHOS RELACIONADOS

Esse Capítulo apresenta trabalhos relacionados e introduz algumas iniciativas de pesquisa desenvolvidas no escopo das atividades de software para sistemas espaciais do INPE e trabalhos realizados fora do INPE com objetivo de comparar estes trabalhos externos com os processos dos satélites NanosatC-BR2 e Amazonia-1.

4.1 Trabalhos relacionados com contribuições para o INPE

No âmbito da produção de software para sistemas espaciais, a Engenharia de Software desempenha um papel fundamental, abordada na norma ECSS-E-ST-40C. Conforme apresentado na Figura 2.1, os processos relacionados a software no padrão ECSS são classificados como (i) processos do ciclo de vida e (ii) processo de apoio ao ciclo de vida. As atividades realizadas nos processos do ciclo de vida abrangem desde a concepção e especificação até a entrega final, evidenciando a complexidade e a interdependência das fases de desenvolvimento. As atividades de Garantia da Qualidade de Software são classificadas como apoio aos processos do ciclo de vida. As saídas geradas pelas atividades realizadas ao longo dos processos (i) e (ii) são denominadas Artefatos de Projeto. Os artefatos de projeto são elementos chave para a Garantia da Qualidade de software. Materializam as entregas nos marcos de projeto. Constituem a base para a equipe da qualidade realizar as atividades de verificação e validação, em geral por meio de revisões e de testes.

Os trabalhos científicos apresentados a seguir representam contribuições para a melhoria do processo de desenvolvimento de software no INPE, uma vez que abordam a complexidade inerente a esse processo, muitas vezes com uma equipe reduzida, prazos apertados e requisitos orientados por uma padronização rigorosa e cumprimento de padrões de qualidade espacial.

O trabalho de Mattiello-Francisco et al. (2007) explora a adaptação de requisitos das normas ECSS para fornecedores de software espacial. A inserção da indústria no ambiente dos programas espaciais como fornecedora de subsistemas de software para satélites exige melhorias nos processos de gestão de projetos para que os projetos sejam concluídos com sucesso. Em diferentes áreas de aplicação, iniciativas, tanto industriais quanto governamentais, podem ser observadas em direção à padronização e ao uso das

melhores práticas para a gestão de projetos, como por exemplo a Cooperação Europeia para Padronização Espacial – ECSS que resultou de esforços cooperativos entre a Agência Espacial Europeia – ESA. Com foco na garantia da qualidade do software, o artigo apresenta um estudo de adaptação das normas ECSS aos processos de desenvolvimento e software implantados em um fornecedor, que adota o modelo Fábrica de Software, para prestar serviço para o INPE. O estudo analisa os processos implantados na Fábrica de Software e sua aderência aos requisitos do INPE, tendo como alvo o projeto de desenvolvimento de um software OBDH embarcado em balão estratosférico. A adoção do padrão ECSS ao longo do projeto não apenas garantiu a conformidade com os requisitos do cliente, mas também promoveu melhorias nos processos de desenvolvimento de software do fornecedor. Os benefícios proporcionados ao fornecedor, em termos de aprimoramentos de processo, evidenciam que a indústria nacional de software está preparada para atender demandas de fornecimento de software embarcado para o setor espacial. (MATTIELLO-FRANCISCO et al., 2007).

Avanços podem ser observados em pesquisas na área de geração automática de testes métodos de teste baseados em modelos (AMBROSIO et al, 2006; MATTIELLO-FRANCISCO et al, 2012; PONTES et al., 2014; SILVA et al., 2018); formalismos de modelagem (PINHEIRO et al., 2014; ALMEIDA, MATTIELLO-FRANCISCO, 2017); e arquiteturas abertas para apoiar a execução de casos de teste (CONCEIÇÃO et al., 2019; BATISTA, et al., 2018).

O artigo de Batista (2020), demonstra que investimentos em métodos e ferramentas de testes contribuem para reduzir o ciclo de desenvolvimento de software embarcado em missões CubeSats. A padronização da estrutura e das interfaces da arquitetura CubeSat impactam o processo de V&V dos Sistemas Intensivos em software embarcado nos nanossatélites baseados em CubeSats. Esses satélites são compostos por componentes *Commercial Off-The-Shelf* (COTS) e seguem estruturas e interfaces padronizadas, o que permite a redução do ciclo de desenvolvimento de missões. Visto que o processo de V&V, utilizado no INPE, aplicado em projetos de nanossatélites seguem o padrão CubeSat. Sendo assim a sua estrutura ou mesmo os subsistemas desses nanossatélites são totalmente adquiridos como COTS. Portanto, o processo de V&V rigoroso, usualmente adotado em satélites tradicionais, conforme recomendado pelas normas ECSS, é adaptado para refletir a reduzida complexidade no desenvolvimento da plataforma do satélite.

Entretanto, tal redução de complexidade não se aplica aos sistemas de software embarcado nas plataformas CubeSats, apresentando-se um grande desafio para os processos de desenvolvimento e de V&V de software embarcado em CubeSats. O artigo reporta a estratégia adotada pelo INPE no desenvolvimento do software OBDH embarcado no NanosatC-BR2. A estratégia explorou, por meio de testes de integração e uso de modelos, antecipar no ciclo de desenvolvimento a detecção de falhas de interoperabilidade entre o software OBDH e as cargas úteis do satélite com as quais o OBDH se comunica. O formalismo da modelagem comportamental dos softwares comunicantes permitiu gerar casos de testes automaticamente a partir dos modelos de interoperabilidade, seguindo a metodologia InRob (MATTIELLO-FRANCISCO et al, 2012). A padronização, sistematização e automatização dos demais. O desenvolvimento de ambientes de teste padronizados usando métodos e ferramentas baseados em modelo permitiu estabelecer um processo de V&V apoiado por um sistema de teste capaz de validar requisitos do software por meio da execução de modelos, o que contribui para mitigar o problema da falta de experiencia (maturidade) das equipes de desenvolvimento de software para lidar com a volatilidade dos requisitos. A familiarização da equipe com questões de interoperabilidade de forma antecipada no ciclo de desenvolvimento, e a oportunidade de perceber as consequências de mudanças de requisitos antes de implementá-los reduzem custo e tempo no processo de desenvolvimento.

Uma seleção criteriosa de um framework adequado para o desenvolvimento de software de voo é apresentada no trabalho de Miranda (2019). Motivado por um problema real no desenvolvimento de um software de voo de missão de um nanossatélite comercial, com restrição de custo e cronograma, afirma que “Software de voo é um tema complexo, demandando um time de desenvolvimento de software com competências nas áreas de sistemas embarcados, sistemas real-time, engenharia de satélites e operações para poder conceber e realizar um projeto nessa área”.É fato que tais competências são raramente encontradas de forma integrada em uma única equipe de desenvolvimento, em especial em equipes reduzidas. Diante dessa realidade, sua contribuição foi uma abordagem onde delineia quatro etapas essenciais: (i) a seleção criteriosa de um *framework* adequado para o desenvolvimento de software de voo, (ii) a análise minuciosa da conformidade desse *framework* com as normas espaciais vigentes, (iii) a definição de diretrizes de projeto destinadas a otimizar a adesão ao *framework* escolhido e, por fim, (iv) a implementação

de uma ferramenta concebida para facilitar a aplicação dessas diretrizes. O framework selecionado para este estudo é o NASA CFS (*Core Flight System*), cuja eficácia já foi comprovada em diversas missões conduzidas pela NASA desde 2015 (MIRANDA et al., 2019).

Outra contribuição importante para o processo de desenvolvimento de software de satélites é descrita na dissertação de Diniz et al. (2019) que propôs uma abordagem para seleção de processos aplicável a software crítico no VSE e analisou as tendências no desenvolvimento de software crítico em Empresas de Pequeno Porte, do inglês, *Very Small Entities (VSE)* ao longo das últimas duas décadas, identificando as práticas empregadas para adaptar padrões e modelos a projetos de software. Na revisão das publicações foi evidenciado que os critérios de adaptação devem levar em conta as particularidades do projeto para determinar quais processos devem ser realizados. Além disso, os métodos para selecionar critérios e processos são diversos, sendo responsabilidade da organização de desenvolvimento definir como implementá-los. A partir da pesquisa revisada, torna-se evidente que os processos padrão são amplamente praticados e profundamente integrados nas organizações de desenvolvimento. Paralelamente, observa-se que o campo da adaptação de processos é multifacetado e continua a ser objeto de estudo e análise para otimizar a qualidade do produto. A adaptação de processos de software crítico em VSE permanece uma questão em aberto, como indicado pela escassez de pesquisas nesse domínio específico. Esse tema é de grande importância, pois está centralizado na melhoria das VSEs e na demonstração da qualidade de seus processos e produtos, indicando, por conseguinte, o potencial dos processos de VSE nos projetos de software crítico. A comparação entre os padrões de software crítico e os de VSE revelou semelhanças significativas, oferecendo oportunidades para sua utilização complementar. Portanto, a seleção de critérios dos projetos constitui um meio para compreender os fatores de influência nos projetos de software crítico no contexto de VSE e para desenvolver uma noção adequada de adaptação. Uma abordagem sistemática para a adaptação de processos pode ser benéfica no contexto de VSE, onde a estimativa baseada em equipe de especialistas é comum, a documentação é escassa e novos membros da equipe podem não estar familiarizados com todas as atividades e fatores relevantes durante a estimativa. Frequentemente, a adaptação de processos é realizada de forma informal em VSE, e a falta de uma abordagem

documentada pode resultar na perda de experiências valiosas de projetos anteriores. Portanto, são necessários estudos adicionais sobre o uso de perfis apropriados, que abranjam conjuntos de processos simplificados e flexíveis, adaptados à avaliação de cada projeto de software, proporcionando evidências sobre sua viabilidade, completude e aplicabilidade. (DINIZ et al., 2019).

Uma abordagem sistemática de testes de software embarcado em nanosatelites com foco em falhas de interoperabilidade foi explorada no artigo do sistema de teste de arquitetura escalável (SATS) baseado em placas de computador Arduino é proposto em (CONCEIÇÃO et al., 2019) para sistematizar cenários de teste com base nos conceitos MIL (model-in-the-loop) e HIL (hardware-in-the-loop). A abordagem combina testes baseados em modelos (MBT) com técnicas de modelagem comportamental do software, referidas em inglês por *Model-Driven Engineering* (MDE) para apoiar: (i) especificação de requisitos de interoperabilidade, cujo comportamento é modelado em máquinas de estado; (ii) geração automatizada de código de software a partir de esses modelos serão incorporados em placas de computador Arduino; (iii) geração automatizada de casos de teste nominais com foco nas propriedades de interoperabilidade; (iv) extensão de modelos com 1754 aspectos de robustez; e finalmente (v) geração automatizada de casos de teste com foco em propriedades de robustez. Além disso, uma estrutura de mecanismo de emulador de falha (FEM) é proposta para testes de robustez de software interoperável intensivo subsistemas nanossatélites a bordo. O FEM atua no canal de comunicação fazendo parte da bancada de testes de integração em duas fases do projeto de nanossatélites: (i) requisito de robustez especificação usando modelo no *loop* (MIL) e (ii) robustez validação usando hardware no loop (HIL). Ambas as abordagens foram aplicadas no processo V&V do NanosatC-BR2 (ALMEIDA, MATTIELLO-FRANCISCO, 2017; BATISTA et al., 2019)

4.2 Contribuição de trabalhos relacionados fora do INPE

Nessa seção são apresentados primeiramente seis trabalhos encontrados na literatura que podem ser comparados aos processos de desenvolvimento dos satélites NanosatC-Br2 e Amazonia-1. No final da seção uma tabela resume e compara as características dos processos.

A. **Processo de desenvolvimento de software usando método ágil: Ônibus Espacial Demo-2**

Resumo: O sucesso do projeto Demo-2, realizado pela SpaceX e NASA, oferece valiosas lições sob a ótica do gerenciamento de projetos, programas e portfólios (PPM). Geopoliticamente, o projeto de US$ 2,6 bilhões reduz a dependência dos EUA da espaçonave Soyuz (Rússia) para acesso à ISS, fortalecendo sua posição no setor aeroespacial e destacando o papel do setor privado. A complexidade das missões demonstra como projetos desafiadores podem elevar o conhecimento humano. A tecnologia, especialmente o software, é central nesses projetos, evidenciando sua importância na sociedade moderna (PINTO, 2020).

Considerando-se que o software é cada vez mais central, ao combinar conhecimentos de projetos aeroespaciais e desenvolvimento de software, a NASA transformou áreas altamente complexas em algo mais manejável. Em ambientes complexos como o do projeto Demo-2, a escolha do ciclo de vida do projeto (preditivo, híbrido ou ágil) é crucial, pois influencia a visão da equipe sobre as entregas e a forma de trabalho. Para isso sugere-se avaliar:

Cultura: considera a maturidade da equipe, seu nível de engajamento, a confiança mútua e a capacidade de tomar decisões.

Equipe: analisa o tamanho do time, a experiência anterior e o acesso a executivos de alto nível, além do patrocinador do projeto.

Projeto: verifica as taxas de mudança, se o escopo pode ser dividido em entregas parciais que agreguem valor e a criticidade do projeto.

Em **ciclos de vida preditivos**, como no caso do Demo-2, o planejamento é feito no início do projeto, e qualquer mudança de escopo é gerenciada progressivamente, conforme preconizado pelo PMI (*Project Management Institute*). Já em **ciclos de vida adaptativos (ou ágeis**), as entregas são desenvolvidas em múltiplas iterações, com o escopo sendo definido e aprovado no início de cada uma. Isso permite à equipe responder a altos níveis de mudança devido à instabilidade dos requisitos, desde que haja um entendimento compartilhado elevado entre todos os stakeholders. Nesse tocante, a escolha do ciclo de

vida (preditivo, híbrido ou ágil) a ser utilizado no projeto determina a visão que a equipe deverá compartilhar sobre as entregas do projeto e como deve trabalhar.

A instabilidade dos requisitos pode ser observada em termos de definição, rastreamento ao longo do ciclo de vida e dificuldade de explicitação. A escolha entre ciclos de vida preditivos, ágeis ou híbridos posiciona o projeto entre fronteiras mais desejáveis nos campos complicado e complexo, evitando a instabilidade do caos, conforme ensina Ralph Stacey. (PINTO, 2020).

Constatações:

1. Comparação entre abordagens ágeis e preditiva no contexto de projetos complexos, como os da NASA e SpaceX, especialmente em tempos de transformação digital;
2. Escolha do ciclo de vida do projeto por meio de um framework, considerando fatores como cultura, equipe e características do projeto, o que pode ser extremamente útil ao discutir a adequação das metodologias ágeis em diferentes cenários.

Resultou em análise comparativa entre metodologias ágeis e preditivas em projetos complexos, como os da NASA e SpaceX, destacando a importância da transformação digital. Ele explora a gestão de stakeholders em projetos de grande escala e geopolíticos, revelando como a NASA maximiza benefícios logísticos e estratégicos. Além disso, propõe um framework para a escolha do ciclo de vida do projeto, considerando fatores como cultura, equipe e características do projeto, fornecendo uma ferramenta para avaliar a adequação das metodologias ágeis em diversos contextos.

Em comparação a este trabalho de dissertação, o artigo trouxe uma nova opção de métodos ágeis, sendo abordagem ágil caracterizada por ser iterativa, incremental e flexível, permitindo adaptações rápidas às mudanças. É frequentemente usada em ambientes onde os requisitos podem evoluir durante o projeto como foi o caso dos satélites NanosatC-BR2 e Amazonia 1. Já a abordagem preditiva é mais linear e sequencial, com um planejamento detalhado no início e uma execução seguindo um plano, adequada para projetos onde os requisitos são bem compreendidos desde o início. Quanto ao ciclo de vida por meio de framework mencionado, refere-se a uma estrutura

metodológica usada para ajudar na escolha do ciclo de vida mais adequado para um projeto, levando em conta diversos fatores contextuais, que no caso das análises dessa dissertação foi considerado o ciclo de vida ECSS.

B. Processo proposto que escreve o software perfeito: Ônibus Espacial

Resumo: "O artigo discute as práticas de desenvolvimento de software adotadas pelo grupo responsável pelo software dos ônibus espaciais, destacando importantes lições aprendidas.

Primeiramente, enfatiza-se a importância do planejamento detalhado antes mesmo de iniciar a codificação. Cerca de um terço do processo é dedicado a estabelecer minuciosamente todos os requisitos e detalhes do novo código, evitando mudanças posteriores sem acordo mútuo. Isso demonstra que a qualidade do produto está diretamente ligada ao planejamento prévio. Além disso, destaca a valorização de uma rivalidade saudável entre os codificadores e os verificadores. Essa competição leva a um processo de teste rigoroso, onde a busca por erros é incessante. Essa abordagem resulta em uma detecção precoce de problemas, garantindo alta qualidade e confiabilidade do software. Outro ponto importante é o papel dos bancos de dados. Eles registram a história do código e todos os erros já cometidos, permitindo análises detalhadas e prevenção de problemas futuros. Isso reflete uma cultura de aprendizado contínuo e aprimoramento do processo. Além de corrigir os erros, o grupo foca em corrigir o que levou aos erros, evitando culpas individuais e promovendo responsabilidade coletiva pelo processo. A ênfase no planejamento cuidadoso, na rivalidade saudável, na análise constante do processo e na busca pela perfeição desde o início são lições valiosas que podem ser aplicadas em qualquer projeto de software, independentemente do tamanho ou complexidade" (FISHMAN, 2013).

Fishman (2013) descreve que este *software* foi desenvolvido por um grupo de 260 especialistas da empresa *Lockheed Martin*. Este grupo possui o nível 5 de classificação do Instituto de Engenharia de Software (SEI), indicando sofisticação e confiabilidade extremas.

O processo de desenvolvimento desse software recomenda que antes mesmo de escrever uma linha de código, um terço do tempo é para o planejamento detalhado. Todas as

especificações são minuciosamente documentadas, garantindo que todos compreendam exatamente o que o código deve fazer. Em seguida, dividem-se as equipes de codificadores e verificadores, cada uma com objetivos opostos: os desenvolvedores produzem um código livre de erros, enquanto os verificadores tentam encontrar todos os defeitos possíveis. Essa relação competitiva leva à descoberta de 85% dos erros antes dos testes formais começarem. Além disso, mantêm dois grandes bancos de dados: um registra o histórico de cada linha de código, permitindo rastrear todas as alterações e o motivo delas; o outro documenta todos os erros já encontrados e as correções realizadas.

O grupo enfatiza a importância do processo sobre as habilidades individuais e a criatividade, e seu método tem sido altamente eficaz. O texto destaca que essa abordagem, embora rigorosa, não é cara nem exclusiva da engenharia de software e que nada nas especificações é alterado sem o acordo e entendimento de todos. E nenhum codificador altera uma única linha de código sem que as especificações as descrevam cuidadosamente. (FISHMAN, 2013).

Constatações:

1. Importância de um Planejamento Minucioso - A equipe do *on-board shuttle group* dedica um terço do processo de desenvolvimento de software ao planejamento antes mesmo de escrever uma linha de código;
2. Rivalidade Saudável entre Equipes de desenvolvedores e verificadores;
3. Documentação e Histórico Detalhado - A manutenção de bases de dados detalhadas, tanto para o histórico do código quanto para os erros detectados, permite que a equipe compreenda profundamente cada linha de código e as razões de cada alteração;
4. Disciplina e Profissionalismo - A equipe trabalha de maneira metódica e estruturada, respeitando horários e mantendo um equilíbrio saudável entre vida profissional e pessoal;
5. Simulação e Modelagem - A utilização de modelos de simulação para prever erros e ajustar processos com base em dados reais permite à equipe manter altos padrões de qualidade e eficiência.

Essas práticas combinadas formam a base para um desenvolvimento de software robusto e confiável, essencial para aplicações críticas como o Ônibus Espacial. O exemplo do *on-board shuttle group* demonstra como a combinação de planejamento detalhado, rivalidade construtiva, documentação rigorosa, disciplina profissional e técnicas avançadas de simulação pode levar a resultados excepcionais. Em comparação a este trabalho de dissertação, o artigo trata de questões de gestão de equipes nos Itens 1, 2 e 4, não abordados na análise comparativa dos dois satélites estudados nessa dissertação. Essas práticas combinadas mostram como um desenvolvimento de software pode ser conduzido de maneira eficaz e eficiente.

C. Elevada mortalidade infantil e falhas de Cubesats, uso de Testes *Fuzz*: Nanossatélites da série SUCHAI e PlantSat

Resumo: "O sucesso das missões espaciais CubeSats depende da capacidade de operar adequadamente em ambientes rigorosos. O software de voo é crucial nessas missões, porém, muitas falhas estão relacionadas a erros nesse software, resultando em perda de missão. Testes operacionais extensivos são usados para garantir a qualidade, porém, a pressão por agilidade no desenvolvimento pode limitar os testes. Neste estudo, foram desenvolvidas, implementadas e avaliadas técnicas de teste fuzz para acelerar os testes operacionais de CubeSats mantendo sua completude. Usando três novos CubeSats 3U da Universidade do Chile, foram identificados doze bugs não cobertos por estratégias de teste tradicionais em menos de três dias. Essas falhas foram corrigidas em oito sessões de sprint" (GUTIERREZ et al., 2021).

Os resultados indicam que o teste fuzz melhorou a completude dos testes de software de voo, automatizando o processo com mínima interrupção no desenvolvimento. Esta abordagem, testada no SUCHAI, é aplicável a sistemas com arquitetura semelhante. No contexto de melhoria no processo para o sucesso dos cubesats, Gutierrez et al. (2021) expressam que os testes operacionais extensivos são as principais técnicas usadas pelos desenvolvedores do CubeSats para garantir a qualidade do software de voo e evitar tais falhas. As exigências do "Novo Espaço" pressionam para agregar "agilidade" ao desenvolvimento de software, o que poderia limitar a capacidade de teste. Embora técnicas avançadas e benéficas de teste de software sejam encontradas no campo da engenharia de software, as soluções de software CubeSat dependem principalmente de

testes unitários, simulação de software em loop (SILS) e simulação de hardware em loop (HILS). Neste trabalho, técnicas *fuzz* de teste foram desenvolvidas, implementadas e avaliadas como uma forma de agilizar os testes operacionais dos CubeSats, mantendo sua integridade.

O impacto das ferramentas foi avaliado utilizando os três novos CubeSats 3U em desenvolvimento na Universidade do Chile. Identificaram doze bugs não cobertos pelas estratégias clássicas de testes em menos de três dias. Essas falhas foram relatadas, corrigidas e caracterizadas pelos desenvolvedores em oito sessões de sprint. Os resultados indicaram que o teste de fuzz melhorou a integridade do voo testes de software por meio de automação e quase sem interrupção do desenvolvimento.

O teste *fuzz* é uma técnica automatizada de teste de software que consiste em inserir uma entrada aleatória em um programa para descobrir falhas do sistema. A falha de software é definida como uma ocorrência inesperada comportamento do software que dá um resultado diferente do esperado. Existem três tipos principais de falhas de software: perda de serviço, entrega incorreta de serviço e falhas de sistema/dados" (GUTIERREZ et al., 2021).

Diversas técnicas de teste são usadas para avaliar a qualidade do software de voo. No estado da técnica, a técnicas de teste mais relatadas aplicadas a testes de software de vôo de nanossatélites são simulação de *hardware in loop* (HILS) e simulação com software em loop (SILS). As metodologias HILS (*hardware in the loop simulatio*n) e SILS *(software in the loop simulation*) podem otimizar a produção e reduzir custos globais do processo em determinadas situações. "No entanto, até onde sabemos, nem todas as missões CubeSat documentam os procedimentos de testes usados para garantir a qualidade e robustez do *software*. Além disso, em nossa busca por soluções de testes ágeis, não foi encontrado, qualquer relatado de técnicas de teste de software mais avançadas, como testes *fuzz* para missões CubeSat." (GUTIERREZ et al.,2021).

Seguem alguns pontos-chave sobre *fuzz testing*:

Geração Automática de Entradas: O *fuzz testing* gera automaticamente uma grande quantidade de entradas aleatórias para serem processadas pelo software em teste.

Descoberta de Vulnerabilidades: Ao fornecer essas entradas aleatórias, o *fuzz testing* ajuda a identificar comportamentos anômalos, falhas e vulnerabilidades que poderiam não ser detectadas por testes convencionais.

Observação de Comportamento: Durante o *fuzz testing*, o comportamento do software é monitorado para detectar crashes, falhas de segurança, e outros problemas.

Os resultados do trabalho de Gutierrez et al. (2021) indicam que o teste *fuzzing* melhorou a integridade do software de voo dos CubeSats, identificando e corrigindo bugs rapidamente.

Constatações:

1. Testes de software ágeis com o uso de técnicas *fuzz* para agilizar os testes operacionais de software de voo;
2. A aplicação de testes *fuzz* que identificam e corrigem bugs rapidamente ilustram a eficiência das metodologias ágeis na manutenção e melhoria contínua do software.

Em comparação a este trabalho de dissertação, este artigo trouxe uma nova forma de introduzir modos de falha para mitigar os ricos de falhas no sistema, minimizando os retrabalhos de processo.

D. Plataforma automatizada de teste para verificação e validação: Nanosatélite ZA-CUBE-1

Nesse artigo de Zaidi et al. (2019) do Departamento de Engenharia Mecânica e Aeroespacial, Universidade da Flórida e pesquisadores do Instituto de Tecnologia do Sul Africano Francês (F'SATI), é descrita a criação de um fluxo de trabalho de engenharia e garantia de missão em pequena escala.

Resumo: "Das experiências aprendidas na missão ZA-CUBE-1, é relatado o estabelecimento de um grupo de garantia de missão em pequena escala. A primeira, de uma série de ações previstas, é atingir o teste funcional e a verificação, para ciclos de temperatura modestos. Primeiro foi criado um sistema de teste e medição composto por equipamentos legados do projeto espacial anterior, enquanto uma câmara térmica é adquirida como primeira passagem de validação ambiental. O sistema de teste é

conduzido de forma autônoma usando um controlador de software altamente flexível que é a ferramenta executável para a metodologia concebida de ciclo de vida de engenharia de sistemas e garantia da missão. Além de medições elétricas e de temperatura automatizadas, o software foi desenvolvido para acomodar os resultados da Fase B/C por meio de simulações, protótipos virtuais, emulação de cenários de operação com ferramentas de hardware e software de missão – tudo unificado em uma única plataforma. O sistema é explorado na validação de um Transmissor de banda S, economizando tempo e obtendo informações valiosas sobre o desempenho da transmissão em cargas térmicas, o que pode resultar na revisão dos requisitos da missão e impactar os parâmetros do sistema de satélite. O objetivo do trabalho é encurtar a engenharia da missão iterativa em ciclos top-down e bottom-up, através da automação e, em última análise, garantir consistência e rastreabilidade substanciais no fluxo do projeto" (ZAIDI et al., 2019)

As experiências aprendidas na missão ZA-CUBE-1 levaram ao desenvolvimento de uma plataforma automatizada de engenharia de sistemas que conduz à conformidade no projeto, teste e verificação do sistema. A plataforma implementa a metodologia de engenharia de sistemas coordenando diversos elementos do ciclo de vida e incorporando as ferramentas envolvidas. A verificação e validação (V&V) é conseguida através da integração de uma instalação de teste e medição à plataforma. "Com a plataforma, realizamos testes e verificações elétricas e funcionais rápidas dos subsistemas CubeSat e validação térmica em ciclos de -20 graus C a +50 graus C".

A plataforma é automatizada por um aplicativo de software que executa testes de ambiente térmico e funcional e fornece suporte para fluxo de requisitos, definição de sistema, desenvolvimento embarcado e simulações integrando hardware alvo em tempo real. A plataforma é explorada na validação de um subsistema de comunicações em banda S, economizando tempo e obtendo informações valiosas sobre o desempenho da transmissão sob carga térmica (ZAIDI et al., 2019).

Zaidi et al. (2019) apresentam uma plataforma automatizada de teste para verificação e validação (V&V) visando identificar anomalias, caracterizar seu impacto e reduzir custos de um sistema desenvolvimento para missões CubeSat. A plataforma, que faz parte da Engenharia de Sistemas Baseada em Modelos (MBSE), preenche a lacuna entre as fases pós-projeto e pré-qualificações, obtendo informações das fases de exploração do

conceito, definição e projeto. Além disso, um software, chamado Missurance, controla os equipamentos de teste e recebe dados quando os testes são realizados. Portanto, o software pode notificar se os resultados atendem aos requisitos funcionais e de projeto, e a especificação do teste. A plataforma também foi utilizada para verificação funcional e validação térmica de um transmissor. O trabalho se concentrou na interação das partes físicas e virtuais do sistema, os tipos de testes mencionados são principalmente HIL e SILS. O sistema de teste e medição será gradualmente aumentado para hospedar mais equipamentos de teste e recursos de testabilidade no software de teste *Missurance*. O objetivo é avaliar o desempenho funcional e elétrico por meio de testes em nível de camada física de todos os subsistemas e, eventualmente, do EM (modelo de engenharia) integrado do CubeSat. Depois disso, o EM será submetido à caracterização térmica. Foi enfatizada a abordagem de teste funcional e verificação por meio de cobertura agressiva e automatizada, seguida por testes térmicos. A metodologia apresentada é orientada para teste e medição e permite uma abordagem V&V mais rápida. O framework da metodologia visa integrar simulação, emulação, design e desenvolvimento, alcançando assim a co-engenharia (ZAIDI et al.,2019).

Constatações:

1. Automação e Integração na execução dos Testes: A automação dos testes funcionais e térmicos permitiu uma verificação rápida e eficiente dos subsistemas CubeSat, reduzindo o tempo de desenvolvimento. A integração de testes funcionais e térmicos, ilustra como técnicas de engenharia de sistemas podem ser aplicadas para melhorar a qualidade e confiabilidade em projetos de software ágil;
2. Plataforma Integrada de Engenharia de Sistemas: permitiu coordenar diferentes elementos do ciclo de vida do projeto;
3. Metodologia MBSE (Model-Based Systems Engineering): A abordagem baseada em modelos ajudou a preencher a lacuna entre as fases do projeto e as qualificações;
4. Teste Funcional e Verificação por Cobertura: A ênfase em testes funcionais com cobertura e automatizada permitiu identificar anomalias e garantir a conformidade com os requisitos;

5. Co-Engenharia e Integração de Hardware e Software: A metodologia visa integrar simulação, emulação, design e desenvolvimento, promovendo a co-engenharia e garantindo a validade completa da funcionalidade das unidades e do sistema integrado;
6. Teste Iterativo e Não-sequencial: enfatiza testes iterativos e adaptação contínua;
7. Avaliação Gradual do Desempenho: A plataforma permite uma avaliação gradual do desempenho funcional e elétrico, começando pelos subsistemas e avançando para o modelo de engenharia integrado do CubeSat;
8. Objetivo de Garantir Operação em Ambiente Térmico: Além dos testes funcionais, a validação térmica é crucial para garantir a operação adequada do satélite no ambiente espacial.

Em comparação a este trabalho de dissertação este artigo trouxe pontos que descrevem um conjunto de práticas avançadas na engenharia de sistemas, focadas na automação, integração e validação abrangente para garantir que os CubeSats atendam aos requisitos funcionais e operacionais. Isso inclui a utilização de metodologias baseadas em modelos, plataformas integradas para gerenciar o ciclo de vida do projeto, e técnicas de teste rigorosas e iterativas para assegurar a qualidade e confiabilidade do sistema final. Algumas dessas abordagens foram práticas também utilizadas no NanosatC-BR2.

E. Modelo H4ASD e Framework RUP com ECSS: CubeSat 3U Libertad-2 3U

Resumo: “Os países em desenvolvimento que têm realizado missões de Cubesat para fins acadêmicos geralmente não oferecem programas de engenharia aeroespacial em suas universidades. Isso causa dificuldades para engenheiros tradicionais ao utilizarem formalmente diferentes padrões e estruturas para desenvolvimento aeroespacial, como os padrões de cooperação europeia para padronização espacial e análise e design de missão espacial. Uma maneira pela qual engenheiros de software tradicionais podem entender facilmente as estruturas de um framework aeroespacial, a fim de aplicá-lo no desenvolvimento de partes de software de missões Cubesat, é comparando seus elementos mais importantes com os elementos sugeridos por um método mais familiar. Neste artigo, apresentamos um framework híbrido entre o padrão ECSS-E-ST-40C e o Rational Unified Process, que pode ser utilizado por engenheiros de software tradicionais como

um modelo guia para o desenvolvimento de elementos de software em missões acadêmicas de nanossatélites. O modelo integra os processos e documentação sugeridos pelo ECSS-E-ST-40C com as disciplinas, fluxos de trabalho e artefatos sugeridos no Rational Unified Process. Isso simplifica a estrutura do ECSS-E-ST-40C e permite que engenheiros de software tradicionais compreendam facilmente seus elementos de trabalho. O artigo descreve, como estudo de caso, a implementação do modelo híbrido na análise e design do software de monitoramento e controle em solo para operação da missão do satélite Libertad-2, desenvolvida pela Universidade Sergio Arboleda na Colômbia (GONZÁLEZ et al., 2016).

González et al. (2016) propõem uma estrutura híbrida para orientar a modelagem de desenvolvimento de software de missões de nanossatélites em um ambiente acadêmico, apresentando um framework híbrido entre o padrão ECSS-E-ST-40C e o Rational Unified Process (RUP).

O estudo apresentado analisa o uso do modelo H4ASD (Desenvolvimento de software híbrido de 4 atividades) como uma ferramenta metodológica para o desenvolvimento de elementos de software em missões satelitais acadêmicas, utilizando o *RUP (Rational Unified Process)* como base. A escolha do RUP facilita a assimilação dos conceitos de engenharia de software por engenheiros tradicionais, permitindo a integração dos processos e atividades sugeridos pelo padrão ECSS-E-ST-40C. A estruturação sequencial das atividades, mantendo o foco no método iterativo e incremental, ajuda a organizar as tarefas e melhorar continuamente os modelos de análise, design e implementação. A pesquisa sugere que o modelo H4ASD facilita a compreensão e aplicação do ECSS-E-ST-40C por engenheiros de software tradicionais, promovendo uma estrutura de trabalho que apoia o desenvolvimento de software em missões aeroespaciais acadêmicas.

A sigla H4ASD significa "Hybrid 4-Activity Software Development," que em português é traduzido como "Desenvolvimento de Software Híbrido de 4 Atividades."

O H4ASD foi testado e validado através do desenvolvimento de diversos elementos de software para a missão satelital Libertad-2, demonstrando sua eficácia em um contexto real e acadêmico. Em resumo, o modelo H4ASD busca oferecer uma estrutura metodológica clara e eficiente para o desenvolvimento de software em missões satelitais

acadêmicas, integrando práticas tradicionais de engenharia de software com os padrões específicos da engenharia aeroespacial.

Constatações:

1. Características Principais do Modelo H4ASD: Integração de RUP e ECSS-E-ST-40C. O modelo combina o RUP, um modelo iterativo e incremental de desenvolvimento de software, com o padrão ECSS-E-ST-40C, facilitando a compreensão por engenheiros tradicionais;
2. Utilização de UML: Promove o uso da UML para modelagem, facilitando a comunicação e o design do sistema;
3. Aplicação Prática: Validado no desenvolvimento de software para a missão Libertad-2, demonstrando eficácia em um contexto acadêmico e real.

Em comparação a este trabalho de dissertação este artigo trouxe um método que consegue utilizar o RUP como base e integrando elementos do padrão ECSS-E-ST-40C, o H4ASD adapta-se às necessidades específicas dessas missões. A utilização da UML facilita a comunicação e o design do sistema, enquanto a validação prática na missão Libertad-2 demonstra sua eficácia e aplicabilidade, essa abordagem é um pouco diferente do vimos e é um método que podemos aplicar em nossos processos, pois também utilizam a norma padrão da ECSS e método RUP.

F. Projeto de uma ferramenta de engenharia de requisitos: CubeSat OpenOrbiter

Resumo: "A Iniciativa de Desenvolvimento do CubeSat OpenOrbiter está trabalhando para construir um sistema de espaçonave pequena usando os princípios de software de código aberto e de hardware aberto. Algumas considerações importantes no *design* para CubeSats são requisitos não funcionais. A contribuição-chave deste trabalho é apresentar o design de uma ferramenta de engenharia de requisitos que pode ser utilizada para acelerar a elicitação de requisitos e especificação de qualidades no âmbito do sistema, tais como, disponibilidade, desempenho e segurança, com base no uso de cenários de atributos de qualidade. Foi projetada e implementada uma aplicação protótipo que utiliza o conceito de cenários de atributos de qualidade, originalmente proposto pelo Instituto de Engenharia de Software da Carnegie Mellon (SEI). Esses cenários de atributos de

qualidade são equivalentes a cenários e casos de uso da UML. Eles são construídos especificamente para documentar os requisitos não funcionais e utilizam um *template* de cenário de qualidade" (HASSAN et al., 2017).

O desenvolvimento do CubeSat está em andamento na Universidade de Dakota do Norte EUA. A pesquisa apresentada, segundo Hassan et al. (2017) é um passo crítico no avanço da análise, projeto, desenvolvimento e teste, como o CubeSat, usando uma abordagem moderna de engenharia de software. "Um passo fundamental para selecionar e validar a arquitetura de software ideal é a elicitação adequada e documentações precisas dos requisitos do sistema. Um dos atributos de qualidade é que eles podem ser usados para testar a adequação do projeto. A elicitação e especificação de atributos de qualidade é um processo problemático e demorado. Para tanto, um protótipo assistido de requisitos não funcionais foi desenvolvido e aqui apresentado. O protótipo proposto não está completo e possui muitas limitações. Novas variações do software estão sendo consideradas e novos recursos podem ser adicionados para aumentar sua funcionalidade e conveniência"

A implementação meticulosa de uma ferramenta de engenharia de requisitos e sua integração no processo de design e implementação para o projeto CubeSat da OOSDI é importante, pois a comunicabilidade e a documentação adequada de requisitos não funcionais são uma das principais questões no desenvolvimento bem-sucedido de sistemas críticos de missão, como uma espaçonave CubeSat. Atualmente, esses cenários são criados manualmente, um de cada vez, e não podem ser integrados de forma contínua em um esquema de requisitos de projeto compartilhável. Para ajudar os desenvolvedores de espaçonaves, uma aplicação de software protótipo, utilizando o conceito de cenários de atributos de qualidade, foi projetada e implementada. Os cenários de atributos de qualidade são construídos especificamente para documentar requisitos não funcionais usando o modelo de cenários de qualidade.

O sistema protótipo fornece um único repositório no qual os requisitos não funcionais podem ser elicitados, mantidos e acessados pelos interessados. O uso dessa ferramenta deve permitir e simplificar o processo de documentação de requisitos não funcionais e garantir sua verificabilidade e rastreabilidade.

Constatações:

1. Ferramenta de Engenharia de Requisitos: uma ferramenta para a elicitação e especificação de requisitos não funcionais usando cenários de atributos de qualidade oferece um método estruturado para capturar requisitos essenciais, um componente crítico para o sucesso de projetos ágeis;
2. Aproximação moderna de engenharia de software: O artigo demonstra como a metodologia ágil pode ser aplicada em projetos de CubeSat, destacando a importância de requisitos não funcionais na arquitetura de software.

Em resumo, as análises e contribuições de diversos estudos destacam práticas e abordagens críticas para o sucesso de projetos complexos, como os de CubeSats e missões espaciais. A automação e integração de testes, juntamente com metodologias ágeis e preditivas, são essenciais para melhorar a eficiência e a qualidade do desenvolvimento. A gestão eficaz de stakeholders, a escolha adequada do ciclo de vida do projeto, e a elicitação de requisitos não funcionais são fundamentais para garantir a conformidade e a robustez do projeto.

Em comparação a este trabalho de dissertação, este artigo trouxe um estudo que indica a importância de integrar práticas modernas de engenharia de software, como a metodologia ágil, com ferramentas estruturadas para gerenciar requisitos não funcionais em projetos de CubeSat, visando aumentar a eficiência e qualidade do desenvolvimento. As abordagens e práticas recomendadas no artigo reforçam a efetividade das práticas adotadas pela equipe de software do Amazonia-1 para viabilizar a entrega dos artefatos com menos sacrifícios durante o processo de desenvolvimento do software do OBDH.

Comparação entre os projetos

A Tabela 4.1 apresenta uma comparação dos seis projetos descritos anteriormente com os projetos NanosatC-Br2 e Amazonia-1. A comparação é feita em termos dos seguintes tópicos: Maturidade da equipe; Documentação Completa conforme ECSS.
Gestão de Requisitos; Processo de Desenvolvimento do software; Ciclo de Vida conforme ECSS (Revisões e entregas de artefatos); e V&V automatizados. A Tabela 4.1 é organizada da seguinte forma: as colunas 1 e 2 contém o nome do projeto e a

PROJETOS” e “ INSTITUIÇÃO / EMPRESA AUTOR/ANO” dos trabalhos. As colunas de 3 a 8 contém os tópicos de comparação.

As considerações sobre cada tópico são:

- Maturidade da equipe: consideram-se o nível de engajamento, confiança mútua e capacidade de tomada de decisões, flexibilidade e capacidade de adaptação a mudanças, ser multidisciplinar, capaz responder mais eficazmente às variações nos requisitos e prazos;
- Documentação Completa conforme ECSS: considera-se o atendimento a norma ECSS-E-ST-40C;
- Gestão de Requisitos: adoção de uma ferramenta de gestão de requisitos, com rastreabilidade de requisitos, controle de revisão, casos de teste, e resultados de teste, que permita a agilidade do processo de desenvolvimento do software garantindo a qualidade do produto;
- Processo de Desenvolvimento do software: Formas de entrega de requisitos, incremental e iterativa, significa dividir o projeto em pequenas partes que podem ser entregues e revisadas regularmente.;
- Ciclo de Vida conforme ECSS: As revisões do ciclo de vida do satélite servem como marcos para a entrega da documentação do Software, seguindo as recomendações da norma ECSS.
- V&V automatizado: esse tópico inclui: Desenvolvimento Orientado a Testes (TDD); Programação extrema (XP); Método de Desenvolvimento de Sistemas Dinâmicos (DSDM); Estrutura de Soluções Microsoft (MSF); Automação de Testes.

Tabela 4.1 – Trabalhos Relacionados comparados ao Amazonia-1 e NanosatC-BR2 .

TÓPICOS ANALISADOS		**Maturidade da equipe**	**Documentação Completa conforme ECSS**	**Gestão de Requisitos**	**Processo de Desenvolvimento do software**	**Ciclo de Vida conforme ECSS – Revisões e entregas de artefatos**	**V&V automatizado**
PROJETO	**INSTITUIÇÃO / EMPRESA AUTOR ANO**						
Satélite Amazonia-1	INPE **INPE** (2021)	Contou com uma equipe especializado do INPE e Bolsistas treinados.	Atendeu ECSS exceto em 4 documentos. Conforme Tabela 5-2.	Utilização da Ferramenta de gerenciamento de Requisitos JAMA.	Os ciclos de vida adaptativos com entradas de requisitos incrementais.	Seguiu o Ciclo de vida ECSS do Satélite na sua totalidade, porém, com revisões técnicas em paralelo para as entregas de várias versões do software no AIT.	Não houve testes automatizados e nenhum tipo de métodos ágeis.
NanosatC-BR2	Pós-graduação em Engenharia e Tecnologia Espaciais/ INPE e UFSM **Batista et al**. (2020)	Projeto Acadêmico – Universidade Federal de Santa Maria e Pós-graduação do INPE. Contrato de VSE de software para OBDH	Não atendeu na totalidade faltando a maioria. Conforme a Tabela 6-2.	O Artigo não trouxe informações.	Os ciclos de vida adaptativos com entradas de requisitos incrementais.	Seguiu o Ciclo de vida satélite conforme ECSS, exceto nas revisões DDR e QR, com entrega de várias versões do software nas revisões oficiais ECSS. E entregas de versões do software no AIT.	Geração automática de testes a partir de modelos (MBT). Uso de ambiente de teste projetado para apoiar V&V com modelos (MIL) e com integração de hardware (HIL) na fase de prototipação.

continua

Tabela 4.1 – Continuação.

TÓPICOS ANALISADOS		**Maturidade da equipe**	**Documentação Completa conforme ECSS**	**Gestão de Requisitos**	**Processo de Desenvolvimento do software**	**Ciclo de Vida conforme ECSS – Revisões e entregas de artefatos**	**V&V automatizado**
PROJETO	**INSTITUIÇÃO / EMPRESA AUTOR ANO**						
Projeto Demo-2 Ônibus Espacial	NASA e SpaceX **Pinto, A.C.** (2020)	O autor atribui o sucesso do projeto Demo-2 à cultura de uma equipe madura com tamanho adequado aoprojeto.	O Artigo não trouxe informações. Possivelmente a documentação seguiu as normas da NASA.	O Artigo não trouxe informações.	Considera os ciclos de vidas preditivos. Os ciclos de vida adaptativos com entradas de requisitos iterativo.	Ciclo de vida do projeto escolhido por meio de um framework, considerando fatores como cultura, equipe e características do projeto. Útil para adequação das metodologias ágeis em diferentes cenários.	Comparação entre abordagens ágeis e preditiva no contexto de projetos complexos.,
Ônibus espacial (em adição ao conteúdo do Demo-2)	NASA **Fishman, C.** (2013)	O *software* é desenvolvido por 260 especialistas Destacou Rivalidade Saudável entre desenvolvedores e verificadores indicando ser uma equipe madura	O Artigo não trouxe informações. Possivelmente a documentação seguiu as normas da NASA.	O Artigo não trouxe informações sobre ferramentas, somente ressalta sobre a importância de banco de dados, para o caso de codificação do software.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	Utilização de modelos de simulação para prever erros e ajustar processos com base em dados reais visando manter altos padrões de qualidade e eficiência.

continua

Tabela 4.1 – Continuação.

TÓPICOS ANALISADOS		**Maturidade da equipe**	**Documentação Completa conforme ECSS**	**Gestão de Requisitos**	**Processo de Desenvolvimento do software**	**Ciclo de Vida conforme ECSS – Revisões e entregas de artefatos**	**V&V automatizado**
PROJETO	**INSTITUIÇÃO / EMPRESA AUTOR ANO**						
Cubesats, Nanossatélites SUCHAI-I	Universidade do Chile **Gutierrez et al.** (2021)	acadêmicos da Universidade de Chile.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	Técnicas *Fuzz* de teste foram desenvolvidas, implementadas e avaliadas como uma forma de agilizar os testes operacionais dos CubeSats, mantendo sua integridade.
Nanosatélite ZA-CUBE-1	Universidade da Flórida, EUA e *French South African Institute of Technology* **Zaidi et al.** (2019)	acadêmicos Universidade da Flórida, Gainesville, EUA	O Artigo não trouxe informações.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	O Artigo não trouxe informações.	Utilizou uma plataforma denominada Missurance, capaz de executar sequencias de teste automaticamente, para verificação funcional com avaliação de cobertura automatizada e teste térmico.
Cubesats, Nanossatélite Libertad-2	Universidade Sergio Arboleda na Colômbia **González et al,** (2016)	Composta por um pequeno grupo de estudantes e engenheiros acadêmicos da Universidade de Colômbia.	Framework híbrido entre o padrão ECSS-E-ST-40C e o *Rational Unified Process (RUP).*	Modelo H4ASD (Hybrid 4-Activity Software Development) é um framework metodológico desenvolvido para orientar o desenvolvimento de software em missões satelitais acadêmicas.	Método iterativo e incremental	Integração de RUP e ECSS-E-ST-40C: O H4ASD utiliza o RUP, que é um modelo de desenvolvimento de software iterativo e incremental e o padrão ECSS-E-ST-40C.	O Artigo não trouxe informações.

continua

Tabela 4.1 – Conclusão.

TÓPICOS ANALISADOS		**Maturidade da equipe**	**Documentação Completa conforme ECSS**	**Gestão de Requisitos**	**Processo de Desenvolvimento do software**	**Ciclo de Vida conforme ECSS – Revisões e entregas de artefatos**	**V&V automatizado**
PROJETO	**INSTITUIÇÃO / EMPRESA AUTOR ANO**						
Cubesat OpenOrbiter	Universidade de Dakota do Norte EUA **Hassan et al.** (2017)	acadêmicos da Universidade Dakota, Estado Unidos.	O Artigo não trouxe informações.	Ferramenta de engenharia de requisitos que pode ser usada para elicitar e especificar requisitos de qualidade, tais como disponibilidade, desempenho e segurança com base no uso de cenários de atributos de qualidade.	Demonstra como a metodologia ágil pode ser aplicada em projetos de CubeSat. Destaca a importância de requisitos não funcionais na arquitetura de software.	Ferramenta de engenharia de requisitos que pode ser usada para acelerar a elicitação adequada de requisitos e documentações adequada de requisitos.	Automação e integração de testes, juntamente com metodologias ágeis e preditivas

5. SOFTWARE EMBARCADO OBDH – AMAZONIA-1

O Amazonia-1, desenvolvido pelo Instituto Nacional de Pesquisas Espaciais (INPE), é o primeiro satélite de observação da Terra completamente projetado, integrado, testado e operado no Brasil. O satélite foi lançado em fevereiro de 2021 da base de Andhra Pradesh, na Índia, e desde então vem operando conforme o esperado, sem apresentar nenhuma falha.

O software fora originalmente desenvolvido pela empresa Argentina chamada INVAP e entregue ao INPE junto com o computador de OBDH e já havia passado por campanhas de testes de entrega e aprovado. Entretanto, o desenvolvimento do Amazonia-1 se deparou com situações conjunturais, inerentes aos desafios associados ao desenvolvimento de tecnologia espacial pela indústria nacional. Tais situações levaram à necessidade de substituição de parte de seus equipamentos de bordo, o que impactou diretamente no software de supervisão de bordo – em inglês, On-*Board Data Handling*, OBDH, que precisou sofrer grandes modificações e complementações.

As importantes modificações necessárias a esse software levaram à substituição de módulos prontos, com reimplementação e revalidação de um grande conjunto de requisitos, bem como troca em muitas interfaces de comunicação com os equipamentos de outros subsistemas. “O Software, então vigente, em sua versão 3.5.51, foi considerado software herdado. Na medida em que as modificações foram sendo implementadas, testes de regressão foram realizados para determinar se funcionalidades já demonstradas continuavam corretas. Isso, somado à melhoria gradual nas ferramentas de operação por telecomando (TM) e telemetria (TC), fez com que o INPE realizasse muito mais testes sobre o software aplicativo (SW APL) do que o próprio fabricante do OBDH, a INVAP” (INPE, 2021).

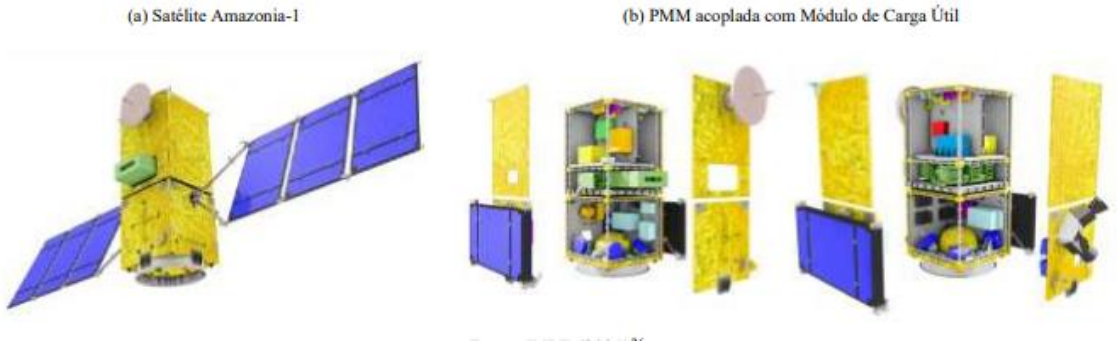
As modificações e a consequente validação foram realizadas por uma equipe pequena de desenvolvedores, dentro de um cronograma estrito, e com fortes restrições materiais, a saber, os novos equipamentos do satélite seriam entregues em etapas avançadas da integração do satélite, e a equipe de software não poderia esperar sua chegada dos novos equipamentos para iniciar o desenvolvimento.

5.1 O Satélite Amazonia-1

O satélite Amazonia-1 é um projeto institucional liderado pelo INPE com participação de múltiplas empresas nacionais e internacionais. Por essa razão, detalhes das implementações do software, regras de operação e de tratamento de falhas, não são descritos aqui.

A Figura 5.1 ilustra a estrutura do Satélite Amazonia-1, sendo os painéis de fechamento separados para demonstrar os equipamentos e subsistemas em sua organização interna, separados em dois módulos. O painel solar fixado ao Módulo de Serviço - tratando-se da Plataforma Multimissão (PMM), localiza-se abaixo na base estrutural, enquanto que na parte superior, encontra-se o módulo de Carga Útil e de seus equipamentos de transmissão e gravação para sensoriamento (WAYRONE et al., 2022).

Figura 5.1 - Vista Aberta da arquitetura do Satélite Amazonia-1: PMM e Carga Útil.

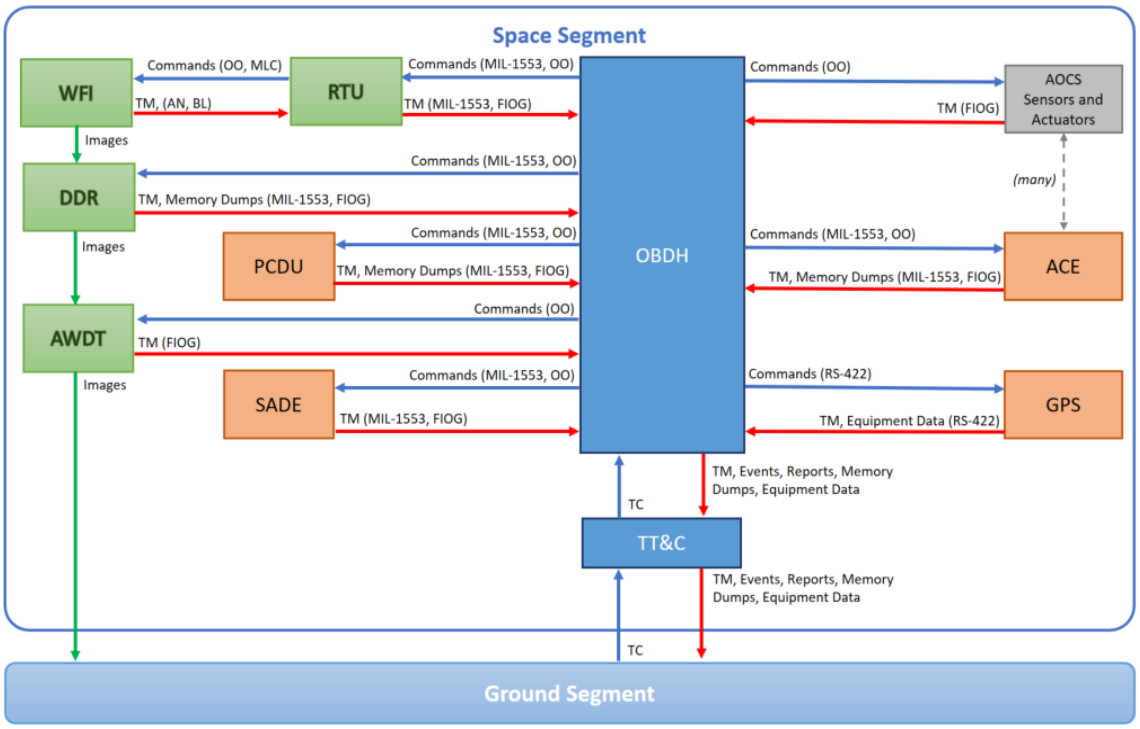


Fonte: Wayrone et al. (2022).

O Amazonia-1 tem como cargas úteis uma câmera óptica de campo largo (‘WFI’) que opera nas bandas do visível e infravermelho próximo, com gravador de dados de imagens (‘DDR’) para aquisições de imagens fora de visada, e um transmissor dedicado para esses dados (‘AWDT’). Para que as cargas úteis do satélite possam ser remotamente operadas, e mantidas em órbita de forma segura, um conjunto de outros equipamentos são necessários, como antenas e transponders para telemetria e telecomandos, painéis solares móveis e baterias para a geração de energia, diversos sensores e atuadores para a determinação e correção do apontamento e órbita. Esses equipamentos fazem parte da chamada plataforma do satélite utilizada na PMM – Plataforma MultiMissão desenvolvida pelo INPE.

O controle e gestão de dados a bordo do satélite Amazonia-1 é realizado pelo subsistema OBDH (do inglês: *On Board Data Handling*). Todos os equipamentos da plataforma do satélite se conectam a um computador central, chamado de Computador do OBDH. A Figura 5.2 traz uma representação esquemática dos principais equipamentos do Amazonia-1, suas principais interfaces e a posição central do OBDH em sua arquitetura.

Figura 5.2– Os principais equipamentos do Amazonia-1 e suas interfaces.



Fonte: Kucinskis (2022).

O software do subsistema OBDH, também chamado de ‘software de supervisão de bordo’, é responsável pelo gerenciamento de todo o satélite, tendo como principais funções: Recepção, análise e execução de telecomandos recebidos de solo; Atuação nos subsistemas do satélite e seus equipamentos; Aquisição de dados dos subsistemas e seus equipamentos; Processamento, formatação e envio de telemetria para solo; Monitoramento dos parâmetros do satélite (diagnose); ‘*Housekeeping*’ (limpeza de buffers, relatos de eventos, auto-testes, entre outros); Gerenciamento, distribuição e sincronização do relógio de bordo; Execução autônoma da sequência de separação do satélite, após a separação do foguete lançador; Gerenciamento dos modos de operação do

satélite; Operação e gerenciamento das cargas úteis; Detecção, diagnóstico e recuperação de falhas (tolerância a falhas); Execução do controle térmico ativo do satélite. Todas essas funções precisam ser executadas de forma concorrente em um hardware qualificado para uso espacial – usualmente de baixíssima capacidade computacional e pouca memória disponível.

O computador do OBDH é baseado em um processador SPARC ERC32 com frequência de clock de 16 MHz, e com apenas 4 MBytes de memória RAM. O software utiliza um sistema operacional multitarefa de tempo real e código-fonte aberto, chamado RTEMS [www.rtems.org], presente em diversas missões espaciais europeias e norte-americanas. Para cumprir todas as suas funções, o software de OBDH do Amazonia-1 é composto por cerca de 30 processos aplicativos concorrentes (KUCINSKIS, 2022).

Quando do início do desenvolvimento do Complemento ao software do OBDH, o satélite Amazonia-1 havia acabado de sofrer importantes mudanças em seu projeto, a saber:

- Substituição de sua carga útil, de uma câmera AWFI para uma WFI acompanhada de uma RTU e seu respectivo conversor DC/DC;
- Substituição de sua PCDU, que também faria as funções de um SADE, por equipamentos distintos, uma PCDU e um SADE;
- Substituição de seu TT&C.

Com exceção do subsistema TT&C, os novos equipamentos eram consideravelmente diferentes dos originais, em termos de interface e operação, para o qual o SW APL já estava preparado.

Os equipamentos, recém adquiridos ou em processo de aquisição, ainda levariam alguns semestres para serem entregues ao INPE, e mais ainda para que modelos deles pudessem ser disponibilizados ao desenvolvimento do software.

Dado o cronograma da missão, não seria possível aguardar a disponibilização dos equipamentos e definição final dos requisitos sistêmicos para iniciar o esforço de modificações ao software já desenvolvido” (INPE, 2021).

5.2 O Processo de desenvolvimento de software espacial e garantia da qualidade

O desenvolvimento de software do OBDH em um projeto espacial é sincronizado com as fases do projeto de satélite e geralmente começa na fase B. Um Plano de Desenvolvimento de Software (PDS) é então preparado para definir um projeto específico e detalhar as atividades de implementação. Os requisitos de software incluem conformidade com satélite, requisitos de subsistemas, suas interfaces, entre outros (ECSS-M-ST-10C, 2009).

O tempo total planejado para o desenvolvimento do SW APL era de aproximadamente 18 meses. Entretanto, devido aos pontos anteriormente destacados, essa previsão se mostrou impraticável. Os atrasos foram significantes conforme demonstrado na Tabela 5.1.

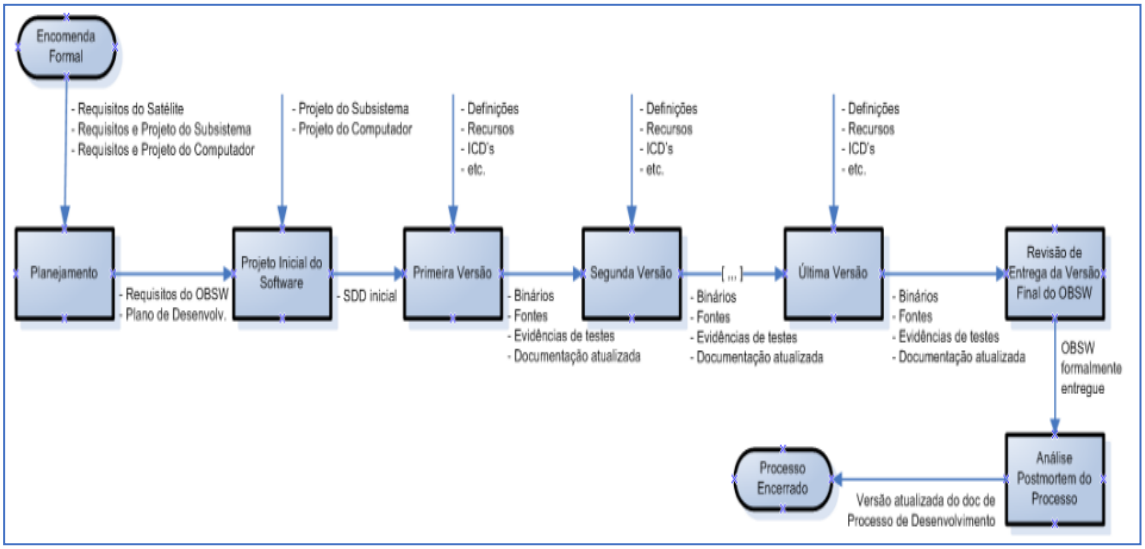
Tabela 5.1 – Cronograma do SW APL: Planejado versus Realizado.

Versão	**Planejado**			**Realizado**			**Atrasos**	**Erro %**
	Início	**Término**	**Dias Plan.**	**Início**	**Término**	**Dias total**		
3.6	18/07/2016	31/08/2016	44	26/08/2016	22/12/2016	118	74	168%
Intervalo entre o término da versão 3.6 e início da 4.0: ***113 dias***								
4.0 (no original, 4.0+5.0)	09/09/2016	30/06/2017	294	04/05/2017	29/03/2018	329	35	11%
Intervalo entre o término da versão 4.0 e início da 5.0: ***71 dias***								
5.0 (no original, 6.0)	07/07/2017	30/11/2017	146	08/06/2018	26/07/2019	413	267	183%
Intervalo entre o término da versão 5.0 e início da 5.1: ***45 dias***								
5.1 (no original, 6.x)	*não aplicável*			09/09/2019	13/12/2019	95	*n/a*	

Fonte: INPE, (2021).

O processo de desenvolvimento de software adotado considera as recomendações do ECSS-E-ST-40C (2009). Este processo apresenta particularidades devido à volatilidade dos requisitos relacionados com alterações de equipamentos numa fase avançada do ciclo de vida do satélite. O processo é iterativo, permitindo alterações de requisitos entre etapas e a identificação de novas versões em cada etapa. Uma visão geral do processo é mostrada na Figura. 5.3.

Figura 5.3 - Visão geral do processo de desenvolvimento.



Fonte: INPE, (2016).

A partir de uma encomenda formal entregue à equipe de especialistas em projetos de software OBDH do INPE, inicia-se a análise dos documentos de Requisitos do Satélite e seus Subsistemas, incluindo os requisitos do subsistema OBDH, para fins de planejamento do projeto de software OBDH.

“O processo de desenvolvimento do software tem como início a encomenda formal oficializado por um documento configurado. Neste documento especifica o projeto, sistema, subsistemas, equipamento e interfaces com os seus respectivos requisitos específicos e o cronograma do satélite” (INPE, 2016).

Após a entrada formal, foi realizado o plano de desenvolvimento e o processo adequado para o desenvolvimento do software. Em seqüência, conforme os requisitos são compilados e entregues ao grupo de desenvolvimento, as versões do software implemetado foram sendo entregues.

Como ilustrado na Figura 5.3 - Visão geral do processo de desenvolvimento, a cada retângulo há entradas de novos requisitos, recursos, ICDs, entre outros, que resultam como saída em atualizações de binários, fontes, testes e documentação atualizada. Uma revisão técnica foi realizada para entrega de cada versão de software, esse processo se repetiu até a versão final ser validada e entregue formalmente por meio de uma revisão de entrega oficial. Após a entrega oficial do Software o processo se encerra com as

análises de Lições Aprendidas.

A cada versão entregue uma revisão técnica de prontidão (RTP) é realizada com os interessados do projeto *(stakeholders*) no intuito de validar as modificações e resultados dos testes de verificação e validação, garantindo que o desenvolvimento do software contempla todos os requisitos previstos do projeto. No caso de alguma discrepância, a versão é corrigida e novamente avaliada pela equipe e assim até oficializar a aceitação de cada versão. Ao final é realizada uma revisão técnica de entrega, para oficializar junto aos interessados a entrega formal da última versão. Junto com cada versão, a documentação de revisão, requisitos e relatórios são entregues para manter a rastreabilidade do desenvolvimento do produto, incluindo as alterações previstas e as não previstas pelo plano, mas que foram necessárias durante as fases do ciclo de vida do projeto.

A cada versão foram acrescentados novos requisitos conforme a demanda correlacionada a novos equipamentos ou a adequação dos recursos adquiridos ou não adquiridos. As propostas de novos requisitos são condicionadas às peculiaridades de cada problema a ser resolvido, sendo por bugs, e falta de equipamento ou na atualização do software em diversos casos como a solução de interfaces durante a AIT. As novas demandas de requisitos em muitos casos foram motivo de atrasos no desenvolvimento do software, por fazerem parte de soluções de projeto ou recursos.

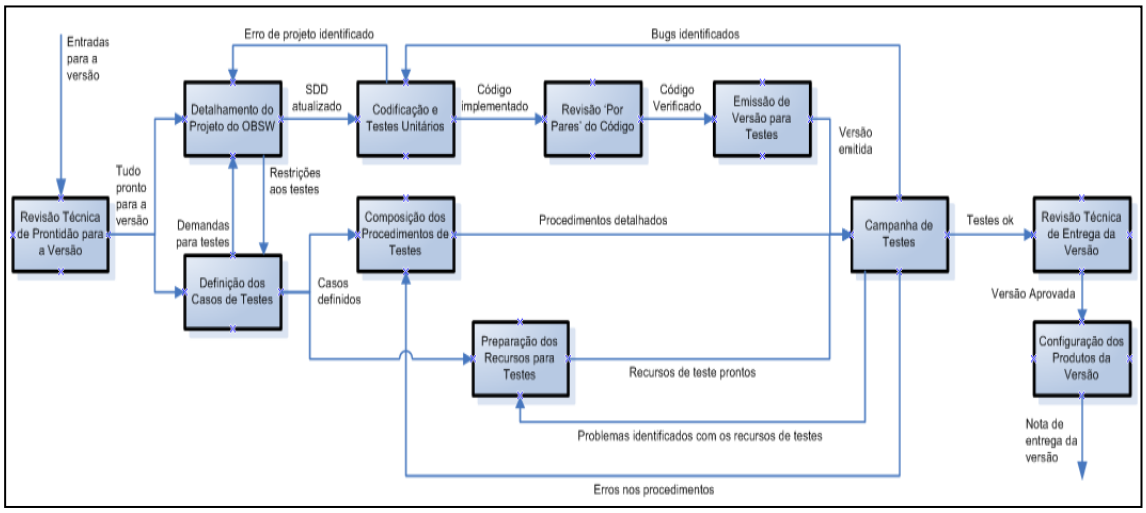
Cada versão a ser implementada tem um ciclo próprio de desenvolvimento, ilustrado na Figura 5.4. Esse ciclo se inicia com uma revisão e definição detalhada do escopo da versão, seguindo para um refinamento do projeto do software, no que diz respeito àquela versão, e então atividades paralelas e interdependentes de codificação e testes. Essas atividades culminam numa campanha de testes de entrega da versão.

As Revisões Técnicas ocorrem ao início (de Prontidão) e final (de Entrega) de um ciclo de desenvolvimento de uma nova versão do software, e permitem acompanhar suficientemente os progressos, discutir e comportar as modificações de requisitos, de escopo e mudanças no cronograma-macro do satélite – como por exemplo, o atraso na entrega de um equipamento. A Figura 5.3 demonstra o ciclo de desenvolvimento interno a cada versão do software.

O processo não estabelece, a priori, uma relação entre as versões incrementais do software

e os marcos da missão, como PDR, CDR e AR. Esta relação é estabelecida na etapa de Planejamento, como as etapas de integração, e dos recursos e informações disponíveis para implementação e teste das funcionalidades exigidas para cada versão.

Figura 5.4 - Ciclo de desenvolvimento interno a uma versão do software.



Fonte: INPE, (2016).

O processo adota uma redução do formalismo de desenvolvimento, a saber: as únicas atividades formais previstas são as Revisões de Requisitos, Planejamento, Projeto e Entrega Final. Não há revisões formais em fases intermédias. Em vez disso, adota-se o conceito de Technical Software Review estabelecido em ECSS-E-ST-40C, 2012), e detalhado em (ECSS-E-HB-40A, 2013).

O processo de desenvolvimento de software é apresentado como um fluxograma detalhado que inclui as seguintes etapas:

1. Ponto de partida:

2. Revisão da preparação da versão técnica, envolvendo:

a) Detalhes do projeto;

b) Codificação e testes unitários;

c) Revisão por pares do código;

d) Emitir versão para teste;

Em paralelo:

a) Definição de casos de teste;

b) Composição dos procedimentos de teste;

c) Preparação de recursos de teste;

d) Campanha de teste.

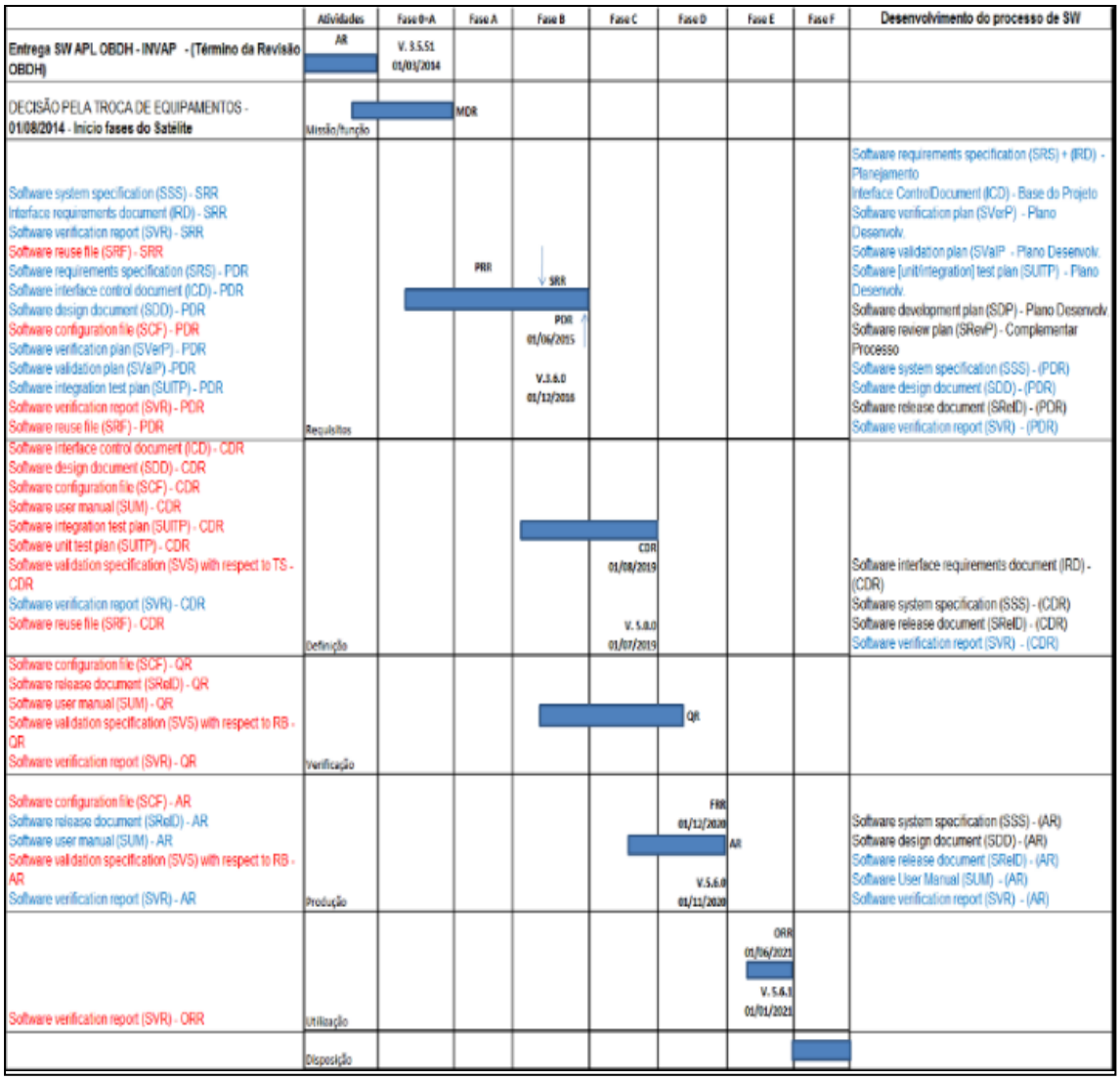
3. Revisão técnica e entrega de liberação e configuração do produto de liberação.

Para cada entrega de versão, uma Revisão de Prontidão Técnica (TRR) é conduzida com as partes interessadas do projeto, principalmente: Arquiteto de Tratamento de Dados, Engenheiro de Sistema, Coordenador do Programa e especialistas de domínio relacionados aos recursos dessa versão (por exemplo, equipe térmica se um algoritmo para o controle térmico deve ser implementado). As partes interessadas validam as modificações e os resultados dos testes, garantindo o cumprimento dos requisitos do projeto. Havendo discrepâncias, a versão é corrigida e reavaliada até ser aceita.

Ao final é realizada uma Technical Delivery Review (TDR) para a entrega formal da versão mais recente. Documentação de revisão, requisitos e relatórios são entregues com cada versão para manter a rastreabilidade do desenvolvimento, incluindo mudanças esperadas e inesperadas durante o ciclo de vida do projeto.

O Projeto Amazonia-1 seguiu a norma ECSS-E-ST-40C (2009) para a entrega de documentos mas, priorizou os documentos mais relevantes, incluindo o conteúdo de alguns que não foram entregues em outros documentos. As revisões do ciclo de vida do satélite serviram como marcos para a entrega da documentação do Software do OBDH, seguindo as recomendações da norma ECSS. As etapas do processo e os produtos gerados foram elencados e comparados com aqueles previstos pelas normas da ECSS, notadamente do ramo de qualidade de software. Uma análise qualitativa foi realizada sobre os documentos “faltantes” no caso do Amazonia-1, e seu impacto sobre o desenvolvimento, sempre sob a óptica de custo versus benefício, em um projeto com equipe reduzida, cronograma restrito e recursos limitados. A análise avaliando os documentos entregáveis no processo de software do Amazonia-1 e sua conformidade com os padrões ECSS é apresentada em (Lima et al., 2022) e está resumida na Tabela 5.2.

Tabela 5.2 – Tabela de comparação de entregas – Projeto Amazonia-1.



Os documentos mencionados na Tabela 5.2 são:

- Na coluna mais à esquerda consta a relação dos documentos sugeridos pela norma ECSS;
- Na coluna mais à direita encontram-se os documentos entregues pela equipe de desenvolvimento do SW APL do OBDH;
- Os documentos descritos em cor vermelha não foram entregues;
- Os documentos descritos em cor azul são os documentos entregues;

- Os documentos em preto, foram documentos entregues, porém não exigida na fase em questão da norma ECSS;
- Foram entregues os documentos considerados mais relevantes pela equipe, e em alguns casos os conteúdos desses não entregues foram inclusos nos documentos entregues.

Alguns documentos não puderam ser entregues pelos motivos a seguir:

- ***Interface Requirements document*** **(IRD)**: não houve um documento específico de requisitos de interface, esses requisitos estavam junto aos requisitos de software "gerais";
- ***Software reuse file*** **(SRF)**: não se aplica, foi um desenvolvimento novo;
- ***Software configuration file*** **(SCF)**: não havia um documento para isso, o controle da configuração foi feito via sistema (Subversion), com referências documentais aos identificadores nesse sistema, quando necessário;
- ***Software verification report*** **(SVR)**: os relatórios de teste com matrizes de rastreabilidade ao final fizeram as vezes desse, pode indicar que o entregamos via "Test *ReportswithTraceability Matrix*", a cada versão entregue;
- ***Software interface controle document*** **(ICD)**: foi feito um documento na INVAP o qual até foi transferido para o formato Amazonia-1, porém não foi entregue no pacote de software (KUCINSK, 2022).

A equipe de desenvolvimento do INPE teve a liberdade de realizar revisões técnicas em paralelo ao ciclo de vida do Satélite para não impactar o processo, possibilitando a criação de múltiplas versões beta do software e ajustes durante a fase de AIT. O desenvolvimento do software exigiu um modelo incremental uma vez que os requisitos de sistema não estavam todos disponíveis no início do desenvolvimento do projeto de software, mas o ambiente de AIT demandava uma versão executável do software mesmo antes da versão final do SW APL estar com todas as funcionalidades completa e validada. A entrega de documentos no prazo foi afetada pela necessidade de inclusão de conteúdo e/ou requisitos em todos os documentos entregues.

Outra característica do processo de desenvolvimento adotado para o SW APL é que ele explicitamente fez uso de ferramentas para o controle de modificação de documentos e código-fonte (para esse caso foi adotada a Subversion). Além dessa, foi adotada uma ferramenta para a gestão de requisitos, casos de teste, registros de evidências de teste e a rastreabilidade entre eles, a JAMA (INPE, 2016).

A adoção da ferramenta JAMA exigiu uma aprovação da Garantia da Qualidade do INPE que analisou seus recursos, com especial foco nas evidências, na rastreabilidade e na visibilidade dos históricos de modificações – de documentos, de código-fonte, de requisitos e casos de teste – havendo por fim declarado serem suficientes para permitir um relaxamento na documentação entregável para o software.

O uso da ferramenta JAMA permitiu à equipe de desenvolvimento focar muito mais no detalhamento dos requisitos, na definição de casos de teste adequados, e finalmente no que deveria ser entregue ao Programa Amazonia, e menos na documentação "intermediária", que demonstraria o progresso de um software ainda não completo. Tal progresso era relatado de forma sumária, e formalmente aceito pelo Coordenador do Programa, por meio das minutas das Revisões Técnicas de Entregas. Todo o detalhamento sempre esteve disponível para consulta no histórico das ferramentas adotadas.

5.3 Ferramenta JAMA e sua contribuição no processo

A ferramenta Jama, em especial, se mostrou vital para lidar adequadamente com a dinâmica da evolução dos requisitos. O esforço de modificações ao software já implementado no Amazonia-1, quando surgiu a necessidade de substituição de parte de seus equipamentos, foi descrito em apenas doze requisitos de alto nível, estabelecidos pela Coordenação do Programa.

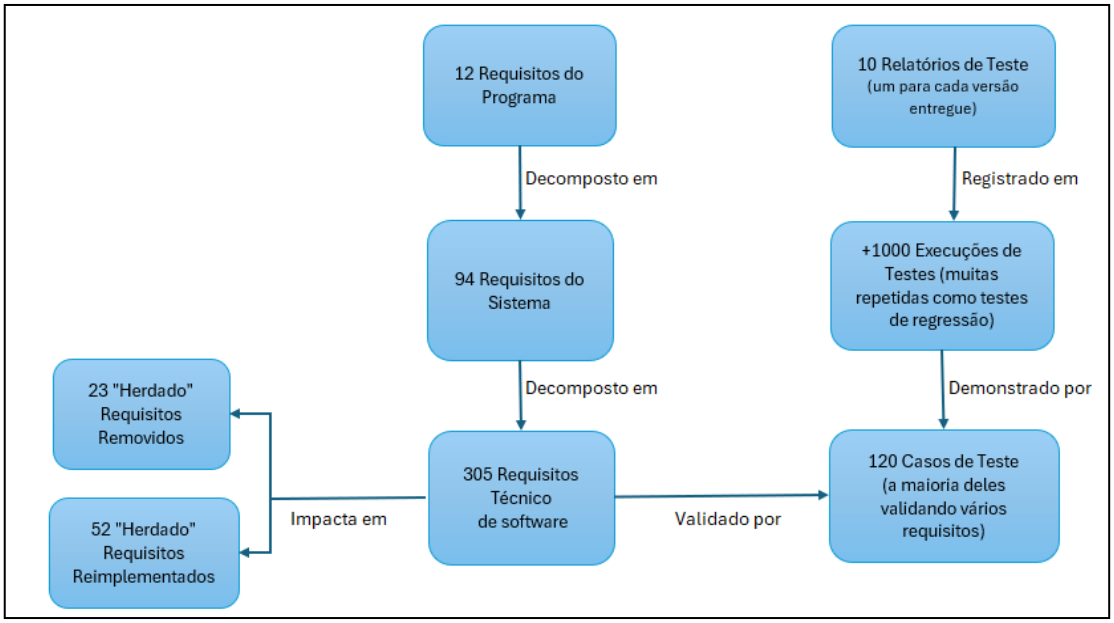
Esses requisitos foram então melhor detalhados pelos Arquitetos de Sistema da missão, em 94 requisitos ditos "sistêmicos". Por fim, os requisitos sistêmicos foram materializados em cerca de 305 requisitos técnicos ao software de supervisão de bordo, com detalhamento e referências suficientes para permitir sua codificação.

Como se tratavam de modificações a um software já implementado, houve também requisitos técnicos removidos (totalizando 23), e outros que foram mantidos, mas reimplementados no código-fonte (52 ao todo), e consequentemente revalidados.

Toda decomposição de requisitos foi realizada diretamente no Jama. Os documentos de requisitos foram tratados como uma “visão” das informações disponíveis no Jama, em uma dada etapa do projeto – como a entrega de uma nova versão do software. Requisitos removidos ou reimplementados também precisaram ser rastreados – afinal, a remoção ou modificação de código-fonte já validado precisa ser devidamente acompanhada e justificada. Em termos de documentação para o Programa Amazonia, era informado apenas quais requisitos seriam trabalhados em uma versão específica do software; o detalhamento e as implicações dessas implementações ou modificações a requisitos estavam disponíveis no Jama e no Subversion.

Além da decomposição de requisitos de Programa, em requisitos sistêmico e finalmente em requisitos Técnicos, o Jama também mantem a rastreabilidade desses últimos para os Casos de Teste. Apartir dos Casos de Teste para os Relatórios de Teste,. Isso permitia a realização de análises de cobertura dos testes (cobertura em termos de funcionalidades, não de código), incluindo os testes de regressão, cuja execução era obrigatória a cada nova versão, para se garantir que nenhuma funcionalidade já entregue havia sido comprometida pela implementação de modificações e novas funções. A Figura 5.5 ilustra a rastreabilidade dos requisitos no Jama.

Figura 5.5 - Decomposição de requisitos e rastreabilidade do software do OBDH Amazonia-1.



Fonte: INPE, (2016).

O Jama também provê um recurso de “links suspeitos”, que informa aos usuários quando uma modificação a um requisito ou caso de teste vinculado foi feito, chamando a atenção para a eventual necessidade de se revisar outros requisitos ou casos de teste relacionados.

Com relação à documentação entregável, demandada pela Garantia de Produto do Programa Amazonia, além do documento de requisitos técnicos, também os relatórios de testes eram exportados diretamente dos registros presentes no Jama. O tempo investido nessas atividades era basicamente o de formatação de documentos, uma vez que a informação já estava disponível no banco de dados do sistema. A análise de cobertura das funções e dos requisitos "velhos" e "novos" a cada versão, proporcionada pela ferramenta, indicava a garantia da qualidade.

Todo processo de desenvolvimento foi realizado por uma equipe de oito pessoas, incluindo o Arquiteto de Supervisão de Bordo, projetistas, desenvolvedores de software e analistas de testes.

“O processo de desenvolvimento de software aqui apresentado engloba as atividades de planejamento, projeto, codificação, controle de configuração, documentação e testes. Também é tratada a interface entre a equipe de software e grupos externos, como o de Sistema, Garantia de Produto de Software, Configuração e os especialistas responsáveis pelos requisitos dos aplicativos a serem implementados no sistema de bordo. O processo visa o desenvolvimento de software embarcado em satélites e de projetos de P&D relacionados a essa área, considerando as peculiaridades de tal domínio. Software” (INPE, 2016).

O Sistema de Gerenciamento de Requisitos Jama, versão 2016.2, desempenhou um papel fundamental na condução do desenvolvimento de software do satélite Amazonia-1, com o servidor sob responsabilidade do SUBORD, como destacado por Kucinskis, INPE (2016). A ferramenta foi aplicada em todas as fases do processo, abrangendo requisitos, casos de testes e controle de rastreabilidade entre eles. Os registros mantidos no Jama foram essenciais, especialmente para requisitos em constante evolução. Uma tabela contendo uma descrição das atividades do desenvolvimento em cada fase do ciclo de vida do software e das atividades relacionadas ao uso da ferramenta JAMA, elaborada pela equipe de desenvolvimento do software SW APL é apresentada no Anexo A.

6. SOFTWARE EMBARCADO - NANOSATC-BR2

6.1 O Satélite NanosatC-BR2

No Brasil, as instituições aeroespaciais e instituições acadêmicas, como faculdades e institutos de ensino superior vem contribuindo com o interesse em participar nesse setor de desenvolvimento de Cubsats, com objetivos técnico-científicos e em alguns casos como educacional também (MCTI, 2017).

Motivado pelo sucesso do NanosatC-BR1, um Cubesat 1U lançado em 2014, que obteve dados úteis para estudos sobre ciência do clima espacial, o NanosatC-BR2 foi inicialmente planejado para ser lançado em junho de 2017. O NanosatC-BR2 é um CubeSat - 2U (10x10x22,6 cm), com massa inferior a 2 kg, projetado para fins científico-tecnológicos e de capacitação (SCHUCH et al., 2020).

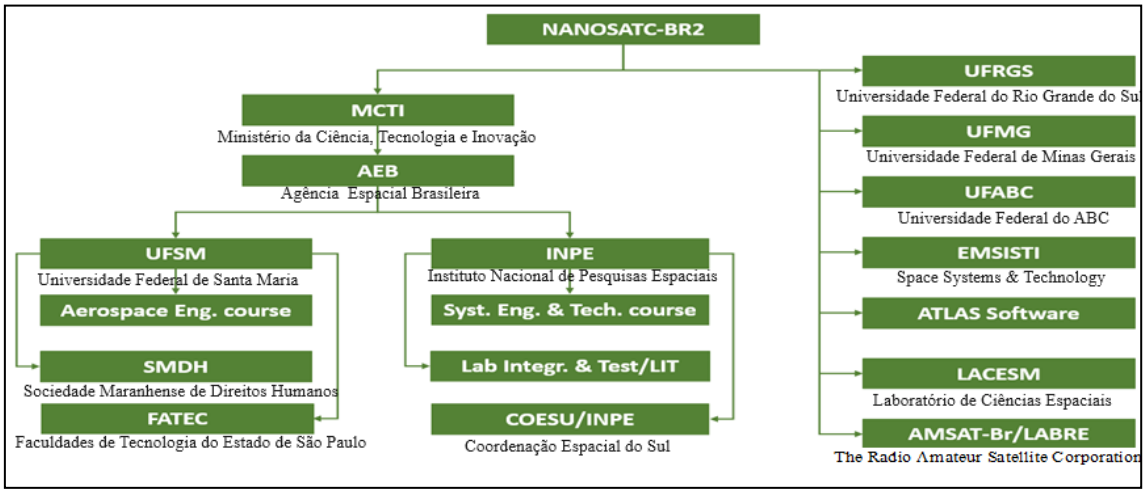
A Figura 6.1 mostra o NanosatC-Br2 com alguns de seus componentes já integrados e uma demonstração do mesmo no espação após ser integrado.

Figura 6.1 – NanosatC-BR2.

Fonte: INPE (2024).

O satélite NanosatC-BR2 foi desenvolvido por um consórcio brasileiro formado por universidades, pequenas empresas e pelo programa de pós-graduação em Engenharia e Tecnologia de Sistemas Espaciais do INPE, conforme mostrado na Figura 6.2 (SCHUCH et al., 2020).

Figura 6.2 - Brazilian Consortium for NanosatC-BR2's Project.



Fonte: Schuch et al. (2020).

Visando atingir o objetivo científico da missão de monitoramento da Ionosfera e do Campo Magnético da Terra, o satélite foi equipado com duas cargas úteis: (i) uma sonda Langmuir desenvolvida pelo INPE, que mede temperatura, densidade e potencial elétrico do plasma e (ii) um sensor de campo magnético XEN-1210 baseado no efeito Hall. Os dados científicos fomentam pesquisas sobre distúrbios na composição da ionosfera na região da Anomalia Magnética do Atlântico Sul (SAMA) no Brasil, que tem graves efeitos nas telecomunicações por satélite, bem como na localização precisa com serviços como o Sistema de Posicionamento Global (GPS). Além disso, foram desenvolvidas nas universidades quatro cargas úteis para qualificação de tecnologias inovadoras em órbita: (i) Sistema de Determinação de Atitude Tolerante a Falhas (SDATF) - uma placa única com um magnetômetro dedicado como um dos sensores e três microprocessadores (em redundância tripla) que incorporam o software de determinação de atitude, desenvolvido pela Un. Federal de Minas Gerais (UFMG) e Un. Federal do ABC (UFABC); (ii) um circuito integrado tolerante à radiação (SMDH) desenvolvido na Santa Maria Design House (UFSM); (iii) um FPGA industrial com software tolerante a falhas desenvolvido pela Un. Federal do Rio Grande do Sul (UFRGS); (iv) um experimento de radiocomunicação amadora (STORE-FORWARD), implementado em software pela AMSAT BR, cujo código-fonte roda no OBC, sob controle do Software On-board Data Handling (referido como OBSw) desenvolvido por alunos do curso de Engenharia de Sistemas e Curso de Tecnologia (PG-ETE) do INPE, São José dos Campos, SP. A Tabela

6.1 resume as características do satélite NanosatC-BR2.

Tabela 6.1 - Resumo – Características e Missão do NanosatC-BR2.

Projeto	Campo de Pesquisa	Instrumento Científico	Instrumentos de Qualificação
NanosatC-BR2 O NanoSatC-Br 2 é um microssatélite brasileiro, ele é o segundo CubeSat da série NanoSatC-Br. NanoSatC-Br 2 é um CubeSat 2U que tem como objetivo duas missões no espaço, uma científica e a outra tecnológica. 	Os dados científicos promovem pesquisas sobre distúrbios na composição da Ionosfera na região da Anomalia Magnética do Atlântico Sul - SAMA sobre o Brasil, impactando severamente as telecomunicações via satélite e a precisão de serviços como o GPS (Schuch et al., 2019a).	Sonda Langmuir desenvolvida pelo INPE (A sonda Langmuir é um dispositivo usado para determinar a temperatura do elétron, a densidade do elétron e o potencial elétrico de um plasma. Funciona inserindo um ou mais eletrodos em um plasma, com potencial elétrico constante ou variável no tempo entre os vários eletrodos ou entre eles e o vaso circundante. As correntes e potenciais medidos neste sistema permitem a determinação das propriedades físicas do plasma).	Sistema de Determinação de Atitude Tolerante a Falhas - SDATF. Circuito integrado tolerante a radiação (SMDH). FPGA industrial com software tolerante a falhas Experimento de comunicação via rádio amador (STORE-FORWARD).

Fonte: Schuch et al. (2020).

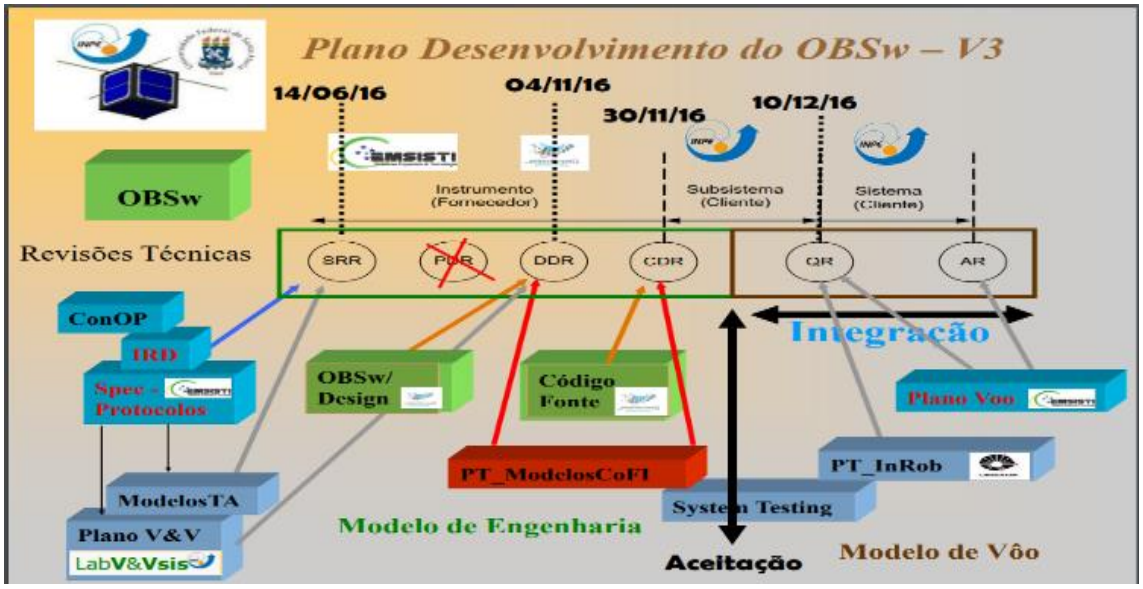
O NanosatC-BR2 foi lançado em 22 de março de 2021 e sua operação em órbita foi realizada pelas Estações Terrestres (UHF e VHF) do INPE em Santa Maria (RS) e Natal (RN) onde os alunos puderam aprender como rastrear e controlar CubeSats em órbita. Além disso, tiveram a oportunidade de obter dados de missões espaciais, de realizar a manutenção e automação de estações terrestres, de receber formação e de preparar documentação sobre uma missão espacial. Após dois meses operando em fase de comissionamento, as informações do farol deixaram de ser atualizadas, perdendo a comunicação com o solo. A falha é atribuída a um efeito de radiação porque o último farol atualizado foi recebido nas Estações Terrestres quando o satélite cruzou a região SAMA.

6.2 O Processo de desenvolvimento de software espacial

A Figura 6.3 sintetiza o Plano de Desenvolvimento do OBSw. No centro da figura estão listadas as Revisões Técnicas preconizadas nas normas ECSS. O processo de software está em conformidade com o processo padrão ECSS em termos de documentos de entrega e revisões. Visando reduzir o ciclo de desenvolvimento, considerou-se excluir a revisão

PDR, justificada pelo fato de o computador de bordo ser uma solução padrão fornecida pelo fabricante da plataforma Cubesat. As caixas inferiores apresentam os conjuntos de documentos objeto de cada revisão. Após a revisão CDR, o código embarcado no OBC seria testado visando concluir a entrega de subsistema. Estava previsto o uso de abordagens *Model-based Testing* (MBT) para a criação dos testes de aceitação, com a metodologia CoFI, que orienta a construção de modelos de conformidade do OBC baseada em serviços (AMBROSIO et al., 2006), bem como, com a abordagem InRob na fase de integração dos subsistemas do NanosatC-Br2 com o subsistema OBC (MATTIELLO-FRANCISCO et al., 2012). Na CoFI os comportamentos normais e excepcionais (casos de falhas) são modelados para cada serviço (HIRATA; AMBROSIO, 2022) enquanto a InRob foca em modelos que representam a interoperabilidae entre os componentes.

Figura 6.3 - OBSw Development Plan.

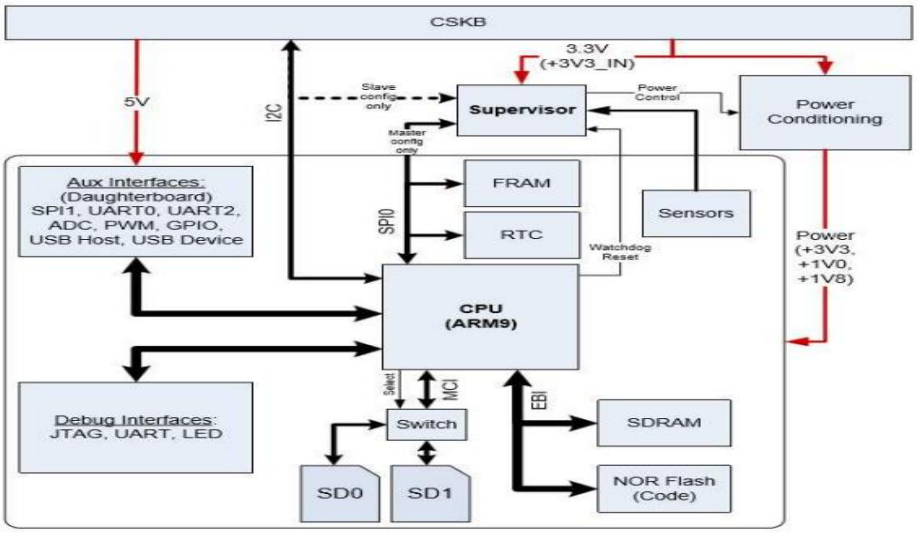


Fonte: Mattiello et al. (2019).

Dado o número e a diversidade de cargas úteis que exigem tratamento e controle de dados OBSw, a maioria delas com baixo nível de maturidade de Nível de Prontidão Tecnológica (TRL), o projeto NanosatC-BR2 enfrentou vários desafios com mudanças de requisitos e falta de informações na Revisão de Requisitos do Sistema (SRR). Além disso, a falta de recursos para o lançamento causou a interrupção do contrato com a pequena empresa contratada para desenvolver o OBSw e o projeto teve que ser reorganizado. O

cronograma foi estendido por 2 anos. O projeto de uma *Daughterboard*, foi desenvolvido no escopo do trabalho de mestrado do INPE para incluir interfaces auxiliares à nova placa do OBC disponibilizada pelo fabricante referenciada por placa iOBC. Os componentes da placa iOBC, incluindo a *Daughterboard* são apresentados dentro do retangulo transparente, no diagrama da Figura 6.1 (ALMEIDA; MATTIELLO-FRANCISCO, 2017).

Figura 6.4 - iOBC block diagram.



Fonte: Almeida e Mattiello-Francisco (2017).

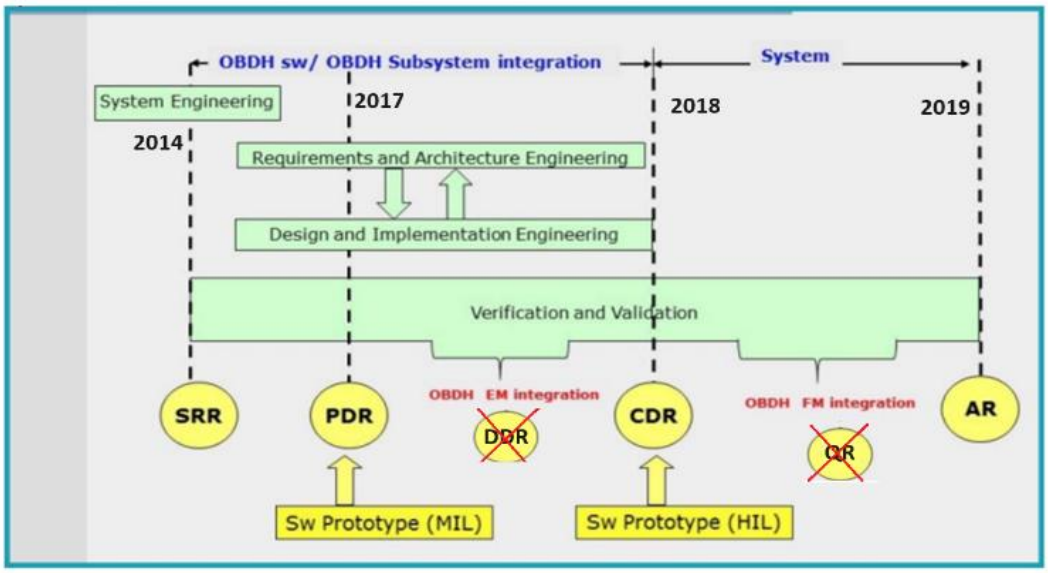
Quando o projeto OBSw foi retomado, uma nova abordagem para o ciclo de vida de Desenvolvimento de Software foi adotada, baseada essencialmente na antecipação dos testes de integração, Figura 6.5, justificada pela pressão do lançamento do satélite em 2019.

Foram eliminadas as revisões DDR - Revisão detalhada do projeto e a QR - Revisão de Qualificação. A justificativa para a primeira é que o DDR visa, em satélites tradicionais, verificar com a equipe contratada para o desenvolvimento, a consistência do projeto dos drivers e bibliotecas de software, porém com o hardware também em desenvolvimento. No caso do uso de plataforma Cubesat, interfaces e protocolos padronizados são disponibilizados pelo fabricante. Em particular, a placa iOBC entregue pelo fabricante com as interfaces auxiliares requeridas garantiu a compatibilidade necessária (BATISTA et al., 2020). A eliminação da QR se justifica pelo uso de uma plataforma de satélite

padrão Cubesat já qualificada. No entanto, a AR - Revisão de Aceitação foi mantida, por se tratar de uma revisão em nível de sistema.

Na Figura 6.5 observa-se também que, diferentemente do Plano de Desenvolvimento inicial apresentado na Figura 6.3, o novo Plano resgata a revisão PDR e elimina a revisão DDR. A justificativa foi o uso da abordagem *Model Driven Engineering (MDE)* que permitiu construir modelos comportamentais do *software* OBSw a partir dos requisitos de sistema, e com isso antecipar a verificação da interoperabilidade de software que se comunicam por meio do barramento I2C. A padronização de hardware nos Cubesats beneficia o projeto de software embarcado devido à disponibilidade já no início do desenvolvimento do software de soluções COTS para o computador de bordo onde o software será embarcado" (BATISTA et al., 2020).

Figura 6.5 – Ciclo de vida de desenvolvimento de *software.*



Fonte: Batista et al. (2020).

A Tabela 6.2 lista os documentos entregáveis no processo e sua conformidade com aqueles estabelecidos nos padrões ECSS, conforme a seguinte simbologia:

- Na coluna mais à esquerda consta a relação dos documentos sugeridos pela norma ECSS;
- Na coluna mais à direita constam os documentos entregues pela equipe de desenvolvimento do software;

- Os documentos descritos em cor vermelha não foram entregues;
- Os documentos em cor azul foram entregues; e
- Os documentos em preto foram documentos entregues, porém não na fase em questão, como exigido pela norma ECSS.

Para cada fase do ciclo de vida, alguns documentos fazem parte do pacote de entrega do software. No Projeto NanosatC-BR2 foram entregues os documentos considerados mais relevantes pela equipe.

Tabela 6.2 – Tabela de Comparação de Entregas – Projeto NanoStaC-Br2.

	Atividades	Fase 0=A	Fase A	Fase B	Fase C	Fase D	Fase E	Fase F	Desenvolvimento do processo de SW
Início do Desenvolvimento do Softwere do NanosatC-BR2	Missão/função		MDR						Não aplicável para o NanosatC-BR2
Software system specification (SSS) - SRR Interface requirements document (IRD) - SRR Software verification report (SVR) - SRR Software reuse file (SRF) - SRR Software requirements specification (SRS) - PDR Software interface control document (ICD) - PDR Software design document (SDD) - PDR Software configuration file (SCF) - PDR Software verification plan (SVerP) - PDR Software validation plan (SValP) -PDR Software integration test plan (SUITP) - PDR Software verification report (SVR) - PDR Software reuse file (SRF) - PDR	Requisitos		PRR	SRR 2016 PDR 2017					Interface requirements document (IRD) - SRR Software requirements specification (SRS) - SRR Software verification plan (SVerP) - SRR Software validation plan (SValP) - SRR Software design document (SDD) - PDR Software Test Plan - PDR Software Test Report - PDR
Software interface control document (ICD) - CDR Software design document (SDD) - CDR Software configuration file (SCF) - CDR Software user manual (SUM) - CDR Software integration test plan (SUITP) - CDR Software unit test plan (SUITP) - CDR Software validation specification (SVS) with respect to TS - CDR Software verification report (SVR) - CDR Software reuse file (SRF) - CDR	Definição				CDR 2018				Software design document (SDD) - CDR Software user manual (SUM) - CDR Software V & V Plan - SRR/CDR Software verification report (SVR) - CDR Misssion Operation Concept - CDR System AIT Plan - CDR Software souource Code - CDR Interface requirements document (IRD) - SRR/CDR Software requirements specification (SRS) - SRR/CDR
Software configuration file (SCF) - QR Software release document (SRelD) - QR Software user manual (SUM) - QR Software validation specification (SVS) with respect to RB - QR Software verification report (SVR) - QR	Verificação					QR			Não aplicável para o NanosatC-BR2
Software configuration file (SCF) - AR Software release document (SRelD) - AR Software user manual (SUM) - AR Software validation specification (SVS) with respect to RB - AR Software verification report (SVR) - AR	Produção					FRR	AR 2020		Software Approval - AR Software User Manual (SUM) - AR System AIT Plan - AR System AIT Report - AR Mission Operation Concept - AR
Software verification report (SVR) - ORR	Utilização						ORR		Não aplicável para o NanosatC-BR2
	Disposição								Não aplicável para o NanosatC-BR2

O processo de desenvolvimento do OBSw é baseado na utilização de modelos: model in-the-loop (MIL) e hardware in-the-loop (HIL), com técnicas de desenvolvimento incremental e métodos ágeis (BATISTA et al., 2020). A abordagem que consiste em uma integração contínua entre OBSw e cargas úteis foi apoiada por contribuições acadêmicas do PG-ETE/INPE desenvolvidas durante 2017-2018. Dois tipos de abordagens baseadas em modelos apoiam o processo: (i) engenharia orientada a modelos (MDE); (ii) Testes

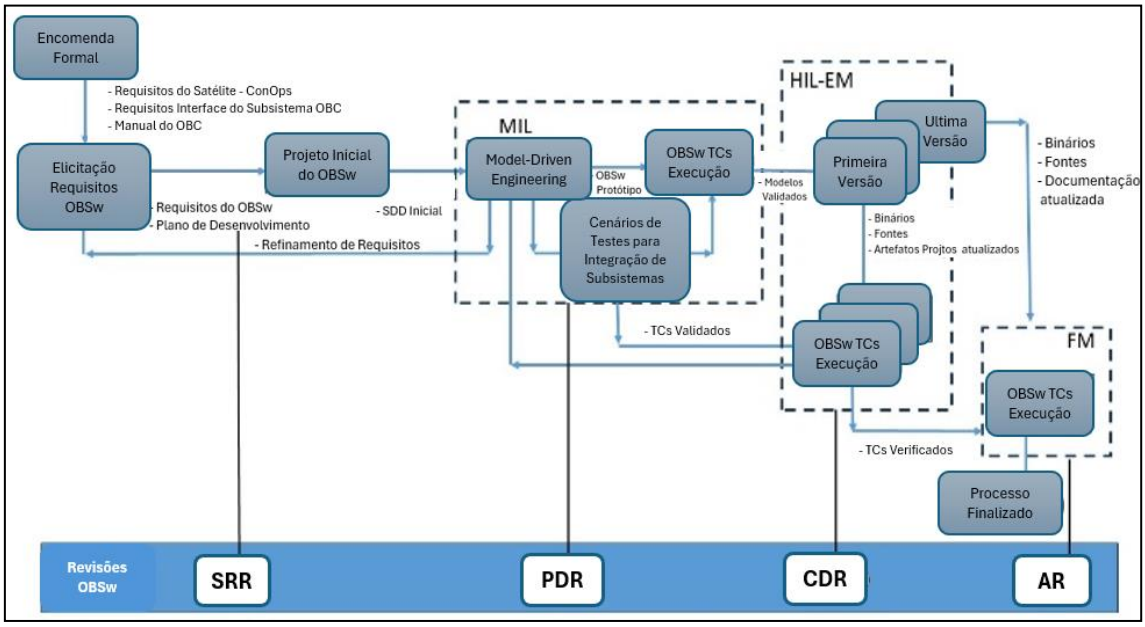
baseados em Modelos (MBT).

A abordagem denominada Scalable Architecture Test System (SATS), (CONCEIÇÃO et al., 2016) permite sistematizar diferentes cenários de testes utilizando MIL e HIL. A abordagem combina MBT e MDE para apoiar: i) especificação de requisitos de interoperabilidade, cujo comportamento é modelado em máquinas de estados; ii) geração automatizada de código de software a partir desses modelos para serem embarcados em placas de computador Arduino; iii) geração automatizada de casos de testes nominais com foco em propriedades de interoperabilidade; iv) extensão de modelos com robustez; e por fim, v) geração automatizada de casos de testes com foco nas propriedades de robustez. O SATS conta com um mecanismo emulador de falhas, denominado FEM (BATISTA et al. 2018) para testes de robustez de subsistemas interoperáveis com uso intensivo de software a bordo de nanossatélites. O FEM atua no canal de comunicação fazendo parte da bancada de testes de integração em duas fases do projeto de nanossatélites: (i) especificação de requisitos de robustez usando MIL e (ii) validação de robustez usando HIL. Permite verificar os requisitos de interoperabilidade e robustez dos subsistemas que interagem através do canal I2C.

O processo de desenvolvimento do OBSw é mostrado na Figura 6.6. Ela destaca as atividades realizadas com o MIL, apoiadas na simulação SATS, e aquelas realizadas com o HIL, utilizando o próprio Modelo de Engenharia (ME).

Vale destacar que o sistema de testes SATS, embora oneroso em termos do esforço necessário para modelar o comportamento dos subsistemas que se comunicam pelo canal I2C, é uma ferramenta extremamente poderosa para verificar se os requisitos de interoperabilidade e robustez entre as partes comunicantes foram corretamente atendidos. Além disso, os cenários de teste derivados automaticamente durante a execução dos modelos são Especificações Abstratas de Casos de Teste, que serão úteis para validar a comunicação dos subsistemas alvo nas fases posteriores do HIL, onde as versões do OBSw já estão embarcadas no iOBC.

Figura 6.6 - Processo de desenvolvimento do OBSw.



Fonte: Batista et al. (2018).

A concepção e implementação do OBSw foram realizadas por uma pequena equipe de uma pequena empresa que contratou ex-alunos de pós-graduação do INPE. As atividades de testes do OBSw e a integração com cargas úteis foram realizadas por alunos do curso de engenharia de sistemas espaciais da PG-ETE (Pós-Graduação em Engenharia e Tecnologia Espaciais) do INPE. Não foi possível estabelecer um processo de verificação e validação conforme recomendado pelo ECSS, com documentação analisada em revisões formais e testes de aceitação do OBSw, devido às constantes mudanças de requisitos exigidas pelos desenvolvedores de *payload*, muitas vezes sem experiência em design de sistemas e com uma equipe altamente rotativa .

Apesar da equipe inexperiente, a fase AIT foi bem sucedidaA razão do sucesso é atribuída à equipe formada por três alunos, sendo dois de mestrado e um de doutorado da engenharia de sistemas do INPE, cujos trabalhos de pesquisa na área de verificação e validação contribuiram com o desenvolvimento de ferramentas e abordagens MBSE capazes de antecipar a detecção de falhas por meio de modelagem e técnicas de simulação.

7. ANÁLISE COMPARATIVA DO PROCESSO DE DESENVOLVIMENTO DO SOFTWARE DOS SATÉLITES NANOSATC-BR2 E AMAZONIA-1

7.1 Análise do processo de desenvolvimento de software do AMAZONIA-1

O desenvolvimento do software OBDH do satélite Amazonia-1 seguiu as fases do ciclo de projeto do satélite, sendo as revisões do software sicronizadas com as entregas de sistema (ECSS-M-ST-10), e não conforme o ciclo de desenvolvimento do Subsistema OBC. Tal excepcionalidade ocorreu porque o software original havia sido inicialmente entregue pela empresa INVAP, caracterizando a entrega de subsistema OBC concluída. Entretanto, após o CDR de sistema, novos equipamentos foram acrescentados ao satélite levando a necessidade de adaptações no software na fase de integração de sistema FASE D. (Tabela 2.2)

O fato das adequações no software OBDH do subsistema OBC ocorrerem na fase de AIT levou à necessidade de estabelecer revisões internas associadas a entregas de versões beta do software, teoricamente encaradas como manutenção do software na perspectiva do ciclo de vida do software. Normalmente, seguindo o ciclo de desenvolvimento do subsistema, após a execução dos testes realizados com o uso de simuladores em laboratório, como parte do processo de verificação e validação do software, espera-se que ocorra a entrega formal e que a operação do OBDH esteja em conformidade com os Requisitos de Base, antes da fase de AIT do satélite. No entanto, o marco de entrega não foi como usualmente feito nas fases de subsistemas, e por conter muitos novos requisitos, foram elaboradas versões betas que atendiam as demandas da fase de AIT do satélite. Isso foi possível por dois motivos: (1) o desenvolvimento do software OBDH ser realizado pelo próprio cliente, com liberdade de fazer revisões técnicas livres (não havendo necessidade de pagamentos estabelecidos em contratos) e, (2) o ciclo de desenvolvimento do subsistema OBC poder ser estendido e parcialmente vinculado ao ciclo de AIT do satélite.

Esse processo teve resultados negativos em termos de cronograma do desenvolvimento e do retrabalho no software, uma vez que após as entregas da revisão CDR, continuou-se o desenvolvimento durante as fases de AIT. Ajustes de bugs e resoluções de problemas das interfaces com os novos equipamentos levaram a realização de 9 versões do software. (INPE, 2016)

A última versão do software aconteceu simultaneamente com a validação do Satélite como um todo seguindo a norma ECSS.

Como os requisitos de subsistema não estavam todos disponíveis no início do desenvolvimento do projeto do software, e o ambiente de AIT demanda uma versão do software antes que o software esteja completamente qualificado, foi possível estender o desenvolvimento do software durante a fase D – Produção e Aceitação (AR) de modo que o software de voo seguiu uma abordagem incremental.

Em termos de documentação associada às entregas do software OBDH, a norma ECSS indica, que para cada fase do desenvolvimento, os documentos devem ser entregues junto com a entrega do software (código fonte). No Projeto Amazonia-1 foram entregues os documentos mais relevantes, considerados pela equipe. Em alguns casos, conteúdos relevantes dos documentos não entregues foram inclusos nos documentos entregues.

No final do processo de desenvolvimento do software foi conduzida uma análise para avaliação de lições aprendidas, identificação de áreas de melhoria e melhores práticas para orientar futuros projetos. A Tabela 7.1 traz um sumário das propostas de melhorias para o processo como uma referência para futuros planejamentos.

Tabela 7.1 – Propostas de melhorias para o desenvolvimento de software OBDH.

Itens	Descrição da proposta de melhoria
1	Incorporar o documento de "Recomendações e Procedimentos Aplicáveis" ao processo, possivelmente na forma de apêndices.
2	Prever no processo que seja dado acesso à base de requisitos, testes e notas de operação (atualmente no Jama) às equipes de AIT e outros usuários do computador de bordo e do subsistema ao qual pertence.
3	Adicionar ao processo um papel específico para o responsável pela manutenção da base de TM e TC, externo à equipe de desenvolvimento do software embarcado.
4	Registrar como recomendação ao processo que as ferramentas de operação adotadas tenham foco em usabilidade na manutenção da base de TM e TC e permitam a injeção de falhas.
5	Adicionar à visão geral do processo de desenvolvimento uma etapa intermediária entre versões, chamada de "Ajustes de Escopo e Provimentos para a Próxima Versão", que funcione como uma interface entre o cronograma do software e o cronograma do projeto no qual está inserido.
6	Adicionar como recomendação ao processo que sejam previstos dois setups completos para o desenvolvimento e testes do software.
7	Prever a figura de requisitos "preliminares" do cliente, a serem inseridos diretamente pelo Engenheiro de Sistemas (ou seu equivalente em um dado Projeto) na ferramenta de gerenciamento de requisitos (atualmente o Jama). Tais requisitos devem ser formalizados junto ao Programa ou Projeto antes da entrega da versão do software em que são implementados.
8	Incorporar ao Processo de Desenvolvimento de Software o Processo de Gerenciamento de Requisitos e do Processo de Verificação.
9	Caracterizar melhor a verificação e validação do software no processo de desenvolvimento.
10	Adicionar ao processo a figura de "pedidos de modificação ao software", diferentes dos requisitos, e sem a conotação dos bugs.
11	Prever como parte do processo a possibilidade de entrega de versões beta do software.
12	Prever ciclos de desenvolvimento menores, com dispensa de RTP (Revisão Técnica de Prontidão) e RTE (Revisão Técnica de Entrega) e campanhas de testes simplificadas para entregas referentes a revisões e correções de bugs do software (incremento nos dois últimos dígitos da identificação da versão).

Fonte: INPE (2021).

O desenvolvimento do software OBDH do satélite Amazonia-1 trouxe valiosas lições que contribuíram para a melhoria contínua dos processos de desenvolvimento de software. As propostas de melhoria visam otimizar a eficiência, usabilidade, comunicação e controle de qualidade dos processos de desenvolvimento e integração, garantindo a entrega de produtos mais robustos e alinhados com as necessidades dos projetos espaciais futuros.

Vale destacar a importância dos Processos de V&V e de Gerenciamento de Requisitos serem incorporados ao Processo de Desenvolvimento de Software, evidenciada nos itens 8 e 9 da Tabela 7.1.

7.2 Análise do processo de desenvolvimento de software do NanosatC-BR2

Conforme descrito na Seção 6.2, o Processo de Desenvolvimento do OBSw do NanosatC-BR2 contou com 4 revisões. As entregas relacionadas a cada revisão do ciclo de vida de

desenvolvimento de software, generalizadas para missões baseadas em Cubesat, estão descritas na Tabela 7.2.

Tabela 7.2 – OBDH Cubesat – Revisões e Entregas da Missão.

Revisões da Missão	Derivados - Satélite Tradicional	Entregas - CubeSats	Responsável pela Entrega
SRR	Linha de base dos requisitos Requisitos de interface Plano de Desenvolvimento de SW Plano de V&V	Linha de base dos requisitos Requisitos de interface Plano de V&V	Cliente Cliente Fornecedor Equipe V&V
PDR	Plano de Desenvolvimento de SW Especificação técnica Plano de V&V	Projeto SW Linha de base dos requisitos Requisitos de interface Plano de Teste SW Relatório de Teste SW	Fornecedor Fornecedor Equipe V&V Cliente Fornecedor Fornecedor
CDR	Projeto SW Código Fonte do SW Plano de Teste SW Relatórios de Teste SW Manual do usuário Plano de V&V Instrumento de Plano de V&V Especificação de V&V Subsistema de Plano de V&V Subsistema de Especificação V&V	Projeto SW Código Fonte do SW Plano de Teste SW Relatórios de Teste SW Manual do usuário Plano de V&V Plano de AIT do Sistema Conceito de Operação da Missão Linha de base dos requisitos Requisitos de interface Aprovação de SW	Fornecedor Fornecedor FornecedorFornecedor Fornecedor Equipe V&VEquipe V&VEquipe V&VEquipe V&VEquipe V&V Cliente Cliente
RA	Aprovação de SW Manual do usuário Sistema de Plano de V&V Sistema de Especificação de V&V Sistema de Relatório de V&V	Manual do usuário Plano de AIT do Sistema Relatório do Sistema AIT Conceito de Operação da Missão	Cliente Fornecedor Equipe V&VEquipe V&VEuipe V&V

Fonte: Batista et al. (2020).

Para cada revisão listada na primeira coluna, estão associados na segunda coluna os documentos entregues, objeto de avaliação nesta revisão em projetos de satélites tradicionais, conforme recomendação ECSS. Na terceira coluna estão relacionados os documentos de entrega final do Cubesat NanosatC-BR2, e na quarta coluna o responsável pela entrega do respectivo documento. Os documentos em negrito destacam a revisão que o documento é efetivamente concluído. Os Requisito de Base e os Requisitos de Interface não são fechados ao final da revisão SRR, como nos satélites tradicionais, mas apenas ao final do CDR (BATISTA et al., 2020). Isso é viável em projetos de software que seguem

abordagens de prototipagem e incrementais.

O fato de o processo de desenvolvimento do OBSw do NanosatC-BR2 ter sido executado por uma equipe pequena, em um ambiente de volatilidade de requisitos, curto ciclo de projeto e baixo orçamento, porém com uma entrega reduzida (mas suficiente) de documentação, demonstra o esforço da equipe em cumprir as normas ECSS. Assim, similaridades com satélites de maior porte são identificadas em termos de documentação entregue – guardadas as devidas proporções – quando as duas Tabelas 5.2 e 6.2 são analisadas.

A própria ECSS reconhece ser uma realidade do desenvolvimento do software embarcado em satélites a dependência do cronograma macro da missão e a definição tardia de requisitos: "O cronograma de desenvolvimento do software de voo é restringido pelo cronograma geral da espaçonave. Requisitos de sistema não estão todos disponíveis no início do desenvolvimento do projeto de software, e o ambiente de AIT demanda uma versão do software antes da qualificação completa do software. Portanto, o software de voo é frequentemente desenvolvido seguindo uma abordagem iterativa." Item 6.2.3.2.2 da (ECSS-E-HB-40A, 2013).

Ou seja, a dificuldade identificada no desenvolvimento do NanosatC-BR2 não é exclusividade do desenvolvimento de pequenos satélites. No entanto, características como requisitos tardios tornam-se mais preocupantes no processo de desenvolvimento de software embarcado em subsistemas dos CubeSats por serem "mais regra do que exceção", o que representa um desafio para os processos de V&V e gerenciamento de requisitos.

7.3 Resultados das análise dos processos de software Amazonia-1 e NanosatC-BR2

Comparando-se os Processos de Software analisados nas Seções 7.1 e 7.2, respectivamente do Amazonia-1 e NanosatC-BR2, similaridades e diferenças foram identificadas em termos de:

a) Maturidade da Equipe:

A diferença entre os dois projetos parece residir em outro ponto apontado por Batista et al. (2020): a maturidade da equipe, ao que podemos associar a existência ou não de um processo de desenvolvimento de software estabelecido e documentado, que comporte

modificações e seja suportado por ferramentas adequadas para a gestão dos requisitos, de testes e da rastreabilidade.

O processo de desenvolvimento adotado com sucesso no Amazonia-1 assume a existência de uma equipe com experiência para sua consecução, apesar de pequena, porém contando com especialistas em diferentes domínios para as análises e detalhamentos necessários aos requisitos. Expressa, enfim, uma maturidade organizacional não usual para projetos de Cubesats com viés educacional.

b) Documentação e Revisões:

Observa-se uma certa similaridade entre os processos de software OBDH dos satélites NanosatC-BR2 e Amazonia-1, no sentido que ambos produziram uma documentação do projeto, não completamente conforme preconizado pelas normas ECSS, porém em ambos consideradas suficientes.

Com relação às revisões, no NanosatC-BR2 o número de revisões formais foi significativamente reduzido. Para a Amazonia1, as revisões técnicas semiformais foram realizadas com mais frequência para monitorar o progresso do desenvolvimento de software e evitar falhas futuras causadas por interpretações erradas. Finalizando, ambos os projetos obtiveram sucesso na fase AIT devido às técnicas e métodos de apoio aos testes adotados nos respectivos processos de V&V. As Figuras 7.1 e 7.2 ilustram os ciclos de desenvolvimento dos softwares OBDH dos satélites Amazônia1 e NanosatC-BR2 respectivamente, destacando as revisões realizadas e os documentos fornecidos.

Figura 7.1 – Ciclo de vida do OBDH do satélite Amazonia-1.

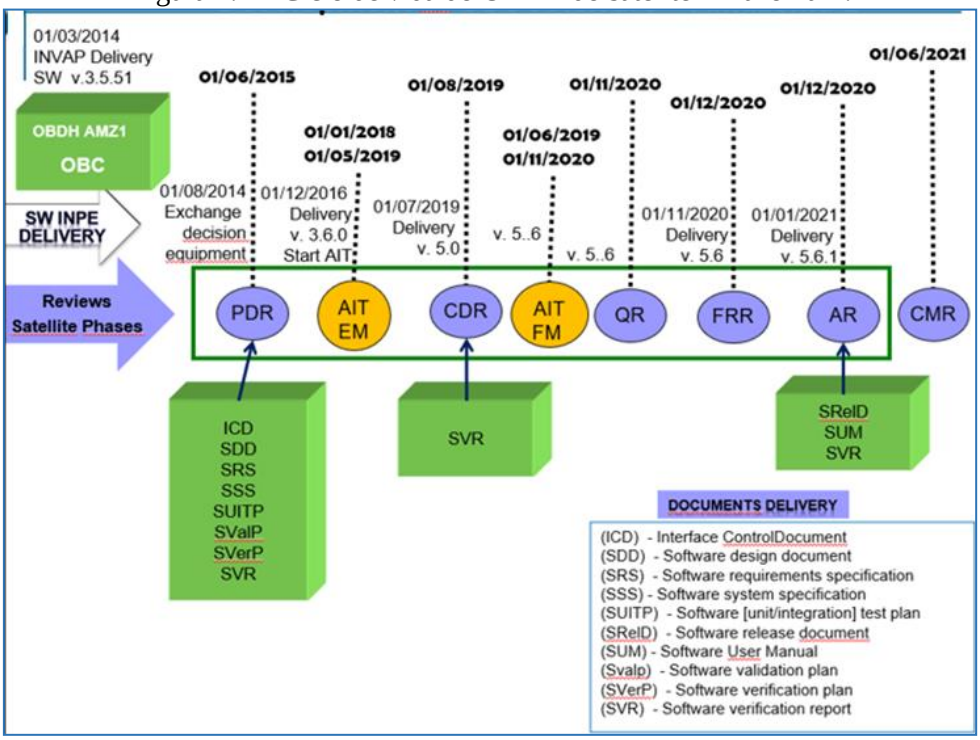
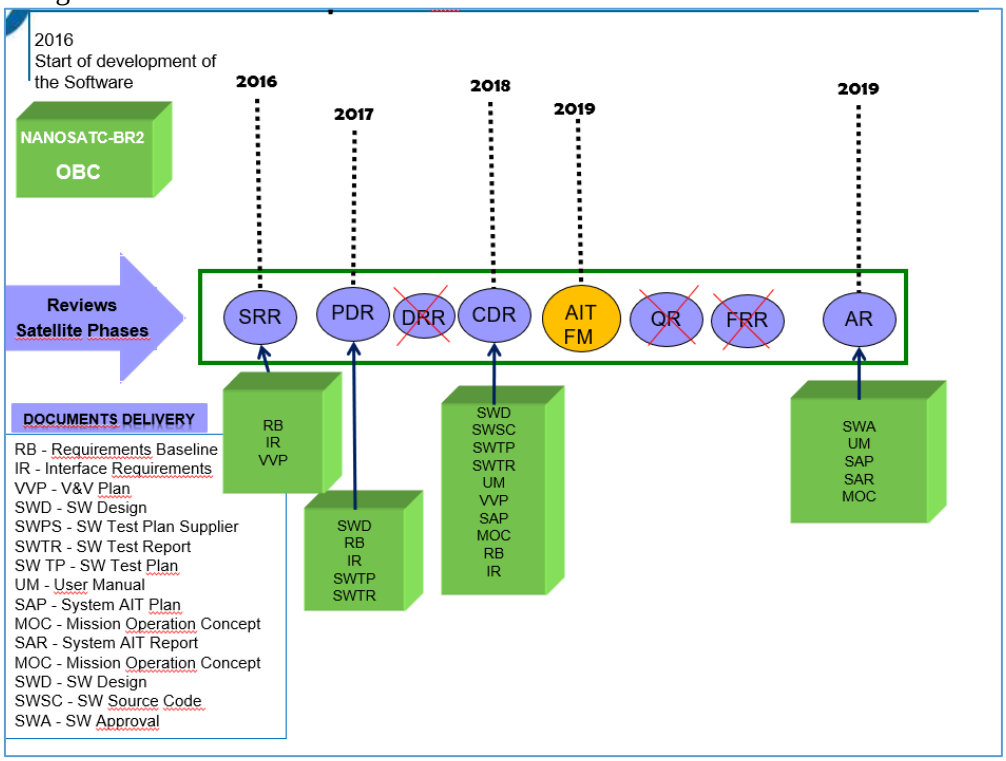


Figura 7.2 - Ciclo de vida do Software do OBC do NanosatC-BR-2.



c) Cronograma:

Em ambos os projetos espaciais há semelhanças no que se refere aos desafios durante a integração, resultando em atrasos e retrabalho de software do subsistema OBC, exigindo uma integração rápida com um software complexo. Nesse ponto, há semelhanças entre o NanosatC-BR2 e um satélite de porte maior.

d) Alteração de Requisitos:

O Amazônia-1 é classificado como satélite de médio porte e o desenvolvimento do Software do OBDH seguiu os padrões ECSS. Porém, enfrentou problemas semelhantes ao NanosatC-BR2 devido ao extenso retrabalho do software para atender às interfaces de novos componentes inseridos ou substituídos em estágio avançado no projeto do satélite. O Amazonia-1 estava em fase de montagem, integração e testes (AIT) quando foi impactado pela substituição de diversos equipamentos. O mesmo aconteceu com o NanosatC-BR2 quando o hardware do computador de bordo foi substituído pelo fabricante. Tanto o NanosatC-BR2 quanto o Amazonia-1 tiveram que lidar com volatilidade de requisitos, entretanto o segundo contou com ferramenta de apoio à gestão de requisitos e de teste, e acompanhamento de um especialista em garantia da qualidade na equipe.

e) Processos de Software:

Os processos de desenvolvimento de software, incluindo as atividades de V&V de ambos os projetos Amazonia-1 e NanosatC-BR2, foram bem estabelecidos e realizados por uma equipe pequena, em um contexto de volatilidade de requisitos, e com redução da documentação formal (mas suficiente) e das revisões previstas pela ECSS. Ambos os projetos foram bem-sucedidos no que tange o desenvolvimento, entrega do software e artefatos, suficientes para qualificação em tempo do lançamento, o que reforça que os esforços investidos em cada processo valeram a pena para garantir a qualidade da missão espacial. Similaridades nos processos de software dos projetos Amazonia-1 e NanosatC-BR2 foram identificadas em termos de “downsizing” do processo original.

f) Testes:

Ambos os processos de desenvolvimento de software adotados no Amazonia-1 e

NanosatC-BR2 fizeram uso extensivo de testes, nas diferentes fases do desenvolvimento, abrangendo testes de unidades, testes de integração e testes de sistemas. *Testes não somente permitiram* **verificar** *se os requisitos de software foram implementados e demais especificações foram contempladas, mas permitiram também a* **validação dos requisitos**, *ou seja, verificar se o sistema atendia às necessidades do usuário e de outras partes interessadas em seu(s) ambiente(s) operacional(is).*

Assim, a técnica de teste é fundamental nos processos de V&V para assegurar que os sistemas cumprem os requisitos especificados e operam conforme esperado em todos os cenários previstos.

A Tabela 7.3 apresenta o resultado da comparação dos processos dos softwares OBDH embarcados no NanosatC-BR2 e Amazonia-1 em termos de similaridades e diferenças, com base nos seguintes critérios: Maturidade da Equipe, Documentação, Revisões, Cronograma, Gestão de requisitos, Processo de software e Teste.

Tabela 7.3 - Comparação das atividades do NanosatC-BR2 e Amazonia-1.

Critérios de Análises	Amazonia-1	NanosatC-BR2	Discussão
Maturidade da Equipe (Diferença)	Oito pessoas entre profissionais com experiência em Satélites e bolsistas orientados pela equipe do INPE.	Cinco pessoas entre estudantes sem experiência com satélites, orientados pela equipe especializada da Pós-Graduação.	O Amazônia-1 contou com uma equipe mais experiente e com especialistas em diferentes domínios, enquanto o NanosatC-BR2 teve uma equipe menos experiente em projetos espaciais, mas ainda assim competente e dedicada.
Documentação Reduzida, mas suficiente (Similaridade)	Atendeu ECSS exceto em quatro documentos. Conforme Tabela 5.2	Não atendeu na totalidade faltando a maioria. Conforme a Tabela 6.2	Ambos os projetos apresentaram uma documentação de projeto incompleta em relação ao padrão ECSS, mas ainda assim, suficiente para cumprir os objetivos do projeto e garantir a qualidade do software.
Gestão de Requisitos (Similaridade)	Uso da ferramenta JAMA garantindo rastreabilidade, casos de testes, relatórios de testes e a vantagem de postergar a data de entrega dos documentos oficiais.	Controlado, rastreado e armazenado manualmente, sem apoio de ferramenta de requisitos.	Tanto o NanosatC-BR2 quanto o Amazônia-1 lidaram com volatilidade de requisitos, especialmente devido a mudanças e substituições de equipamentos durante as fases AIT. O Amazonia-1 manteve controle de requisitos com Jama, enquanto o NanosatC-BR2 seguiu um processo mais ágil, com menos formalidades, e com menos artefatos entregues.
Revisões e cronograma (Similaridade)	Seguiu o Ciclo de vida ECSS do Satélite na sua totalidade, porém, com revisões técnicas em paralelo para as entregas de várias versões do software no AIT.	Seguiu o Ciclo de vida satélite conforme ECSS, exceto nas revisões DDR e QR, com entrega de várias versões do software nas revisões oficiais.	Ambos tiveram dificuldades, com mudanças nos requisitos do sistema, e inclusão de novos equipamentos ou componentes, levando a atrasos no cronograma. O software precisou ser ajustado sendo assim não foi possível acompanhar na integra o ciclo de vida do satélite, isso gerou lacunas nas revisões. Versões do software não puderam ser entregues com uma versão final definitiva nas revisões previstas pela norma ECSS. (foram entregues em revisões beta em reuniões técnicas)
Processo de Software (Similaridade)	Incremental, requisitos entregues com atrasos na fase de AIT	Ágil, requisitos entregues com atrasos na fase de AIT	Ambos enfrentaram desafios semelhantes durante a fase de desenvolvimento, resultando em atrasos e retrabalho devido a alterações em componentes ou requisitos.
Testes (Similaridade)	Testes para verificação, teste de regressão em bancada e testes para validação no ambiente de AIT aplicados a cada versão. Uso de simuladores e equipamentos reais.	Testes gerados automaticamente a partir de modelos, validados em ambiente MIL, e verificados em ambiente HIL-EM. Uso de ferramentas MBT e Sistema de Teste SATS. Processos de V&V estabelecidos.	Ambos os projetos tiveram sucesso na fase de AIT, graças a abordagens de teste eficazes e ferramentas de apoio ao processo de V&V.

Com base nas definições ECSS/IOD apresentadas na Tabela 2.1, um estudo dos ATRIBUTOS de missões Cubesats foi realizado em termos da sua aplicabilidade ao software OBDH. A cada atributo foram associadas práticas da garantia da qualidade do software que pudessem ser aplicadas aos processos de desenvolvimento e processos de verificação e validação de software de Cubesats.

A Tabela 7.4 apresenta o resultado do estudo realizado em 3 etapas:

(i) Identificar como o atributo se aplica ao software OBDH de Cubesats (coluna 3 da Tabela);

(ii) Identificar, quando aplicável, as especificidades do atributo no software OBDH do NanosatC-BR2 (coluna 4 da tabela);

(iii) Identificar as diferenças do atributo no software OBDH do Amazonia-1 (coluna 6 da tabela).

A elaboração da Tabela 7.4 permitiu associar aos ATRIBUTOS, práticas da garantia da qualidade (coluna 5 da Tabela) aplicáveis aos Processos de desenvolvimento e de V&V de software OBDH embarcado em Cubesat conforme descrito a seguir:

(i) Usar ferramenta de Gestão de requisitos, rastreabilidade e testes – facilita o processo de desenvolvimento de software apoiando o acompanhamento mais eficaz de mudanças frequentes de requisitos. **Funções autônomas** implementadas por software aumentam a complexidade dos requisitos especificados. Software mais complexo demanda processos de V&V mais rigorosos;

(ii) Realizar Validação do software em cenários de testes de integração e de sistemas para antecipar situações de risco – facilita a concepção de cenários críticos de operação e especificação de testes para validar o comportamento esperado do software nesses cenários. Ferramentas automáticas que apoiam a especificação dos testes e sua execução de forma automática, aumentam a capacidade de teste e reduzem o tempo de teste. Portanto são compatíveis ao **curto prazo** de projeto;

(iii) Estabelecer documentação mínima verificável – necessário para realizar o desenvolvimento em prazos curtos com equipe reduzida e baixo custo de projeto;

(iv) Estabelecer número mínimo de revisões preconizadas pela ECSS – necessário para realizar o desenvolvimento e

Tabela 7.4 – Aplicabilidade dos ATRIBUTOS ECSS/IOD para software OBDH em missões de Cubesats.

Atributo	Descrição	Aplicabilidade ao software OBDH Cubesats	Especificidades doSoftware NanoSatC-BR2	Práticas recomendáveis garantia da qualidade	Especificidades Software do Amazônia-1
Sistemas Autônomos Completos	Inclui a plataforma, carga útil, segmento terrestre e operações, assegurando que o CubeSat funcione de forma independente.	Aplicáveis em termos de Funções autônomas implementadas por software. Porém, aumentam a complexidade do software	Baixa autonomia do Software -e requisitos instáveis	Usar ferramenta de Gestão de requisitos, rastreabilidade e testes	Autonomia média Software complexo - requisitos alterados - contingência
Maior Perfil de Aceitação de Risco	Aceitação de níveis mais altos de risco em comparação com missões espaciais tradicionais, devido à natureza experimental e de demonstração dos CubeSats.	Aplicáveis, flexibilizando a redução de requisitos de robustez, e de segurança implementados por software	Sem tolerância a falhas, baixa robustez. Aceita desativar operação de payload	Validação em cenários de testes de integração e de sistemas para antecipar situações de risco	Aceitável nível de RISCO baixo
Baixo Nível de Complexidade	Comparativamente menos complexo que outros projetos espaciais da ESA, facilitando o desenvolvimento e implementação rápidos.	Não necessariamente aplicável- depende da missão e quantidade de payloads	Alta complexidade e número elevado de Payloads	Usar ferramenta de Gestão de requisitos e rastreabilidade	Alta complexidade apesar de uma única payload
Baixo Custo	Projetos são desenvolvidos com orçamento limitado.	Aplicável – equipe reduzida e não experiente	Equipe de estudantes	Definir documentação mínima verificável	Equipe experiente. Recurso para contratar equipe de garantia do produto
Curto Prazo	Cronogramas curtos, otimizando recursos e tempo de desenvolvimento.	Aplicável – ciclo de desenvolvimento de software curto	Desenvolvimento ágil, menos revisões, menor # documentos	Definir mínimo de revisões preconizadas pela ECSS	Desenvolvimento Incremental seguindo processo estabelecido INPE

continua

Tabela 7.4 – Continuação.

Atributo	Descrição	Aplicabilidade ao software OBDH Cubesats	Especificidades doSoftware NanoSatC-BR2	Práticas recomendáveis garantia da qualidade	Especificidades Software do Amazônia-1
Ciclo de Vida Operacional Curto	Tempo de operação típico é inferior a um ano, permitindo rápida iteração e validação de tecnologias.	Aplicável – por exemplo, não requer funcionalidades do OBDH para upload de código	# elevado de Payloads dificulta cenários de operação, interação em operação	Validação em cenários de testes de integração e de sistemas para antecipar situações de risco	Funções de upload de código implementadas
Aceitação de Ponto Único de Falha	Reconhecimento de que a falha de um único componente pode levar à falha da missão, aceitando este risco como parte do perfil operacional.	Aplicável	Aceitável desativar operação de payload	Validação em cenários de testes de integração e de sistemas para antecipar situações de risco	Não aceita ponto único de falha
Redundância Limitada e Tolerância a Falhas	Sistemas são projetados com redundância mínima, sendo a tolerância a falhas um aspecto menos prioritário.	Aplicável – poucas FDIR implementadas, sem redundância	Equipe de estudantes	Testes extensivos da arquitetura do software	Muitas FDIR implementadas
Modo de Segurança Robusto	Implementação de modos de segurança para garantir que o CubeSat possa entrar em um estado seguro em caso de falha ou anomalia.	Não necessariamente	Aceitável desativar operação de payload	Validação em cenários de testes de integração e de sistemas para antecipar situações de risco	Aplícável
Uso Extensivo de Elementos COTS	Utilização intensiva de componentes comerciais prontos para uso (COTS), que são mais econômicos e rápidos de integrar.	Aplicável – software OBDH embarcado em OBC -COTS	Bibliotecas e Drivers fornecidos OBC	Testes extensivos da arquitetura do software	Não utiliza COTS – bibliotecas e drivers desenvolvidos in house

continua

Tabela 7.4 – Conclusão.

Atributo	Descrição	Aplicabilidade ao software OBDH Cubesats	Especificidades doSoftware NanoSatC-BR2	Práticas recomendáveis garantia da qualidade	Especificidades Software do Amazônia-1
Testes Extensivos no Nível do Sistema	Realização de testes funcionais e ambientais completos, incluindo qualificação e aceitação, para assegurar o desempenho esperado.	Aplicável – testes funcionais realizados na fase de AIT – Modelo de voo	Retrabalho na fase de AIT	Testes de integração e de sistema	Retrabalho na fase de AIT
Organização Simples do Projeto	Estrutura de projeto simplificada com equipes integradas e poucos fornecedores ou subcontratados, promovendo eficiência na comunicação e execução.	Aplicável - equipe reduzida de desenvolvedores de software, com forte atuação na integração de sistema	Equipe de estudantes e Startup de software contratada	Usar ferramenta de Gestão de requisitos, rastreabilidade e testes	Organização de projeto complexa

(v) Realizar Testes extensivos da arquitetura do software – redundância mínima, sem prioridade para tolerância a falha limitam a capacidade do software de lidar com situações anormais. Testes extensivos da solução arquitetural do software permitem antecipar possíveis falhas de projeto do software. Ferramentas automáticas que apoiam a especificação dos testes e sua execução de forma automática, aumentam a capacidade de teste e reduzem o tempo de teste. Portanto são compatíveis ao curto prazo de projeto;

(vi) Investir em Testes de Integração e de sistema – o uso extensivo de elementos COTS facilitam a solução de subsistemas e sua pronta entrega, entretanto não garantem a compatibilidade entre os subsistemas e sua interoperabilidade. Testes de integração são extremamente importantes para identificar falhas e inconsistências na interação entre os elementos COTS. E testes extensivos no nível de sistema são essenciais para validar cenários operacionais em órbita. Ferramentas automáticas que apoiam a especificação dos testes e sua execução de forma automática, aumentam a capacidade de teste e reduzem o tempo de teste. Portanto são compatíveis ao curto prazo de projeto.

7.4 Lições aprendidas

Boas práticas para garantia da qualidade de software OBDH embarcado em subsistema OBC foram derivadas dos estudos realizados nas Seções 7.2 e 7.3, respectivamente, com base nos seis (6) critérios analisados apresentados na Tabela 7.3 que destacam similaridades e diferenças dos processos adotados no desenvolvimento do software OBDH embarcado no Amazonia-1 e no NanoasatC-BR2, e na análise dos doze (12) ATRIBUTOS da ECSS/IOD, em termos de sua aplicabilidade no componente software OBDH para Cubesats.

Ao longo do processo de análise, seguem algumas lições aprendidas:

- A **complexidade do OBDH** depende dos objetivos da missão. Apesar de o IOD/ECSS assumir que missões CubeSat possuem menos payload e dados para serem processados e manipulados a bordo, na realidade essa não tem sido regra. No caso do NanosatC-BR2, o número de payloads a gerenciar pelo OBDH o torna significativamente complexo, comparável à complexidade do OBDH do

Amazonia-1 que possui uma única payload. Assim **o ATRIBUTO baixo nível de complexidade deve ser perseguido por missões Cubesat**;

- Software complexo exige maior maturidade da equipe de desenvolvimento e, em geral, ciclo de desenvolvimento mais longo. Para manter coerência com equipes **inexperientes** e **ciclos curtos**, o software de missões CubeSat devem ser de baixa complexidade. **Isso implica restringir os objetivos da missão, em Cubesats**;
- Aceitação de risco em missões CubeSat não significa dispensar testes. Trata-se de riscos associados ao uso de COTS, por exemplo, o fato de não serem resistentes a radiação. Em missões de maior porte, COTS não são utilizados e o nível de aceitação de risco é muito menor. Para atingir a exigência de confiabilidade requerida, o custo do projeto torna-se mais alto. Assim, **maior aceitação de risco em missões Cubesat não significa reduzir esforços de testes de software**;
- Testes são recomendados tanto em missões de maior porte quanto em Cubesats, principalmente testes com foco em nível de sistema. Teste é a única técnica de verificação executável, capaz de evidenciar não conformidades de requisitos implementados comparado a sua especificação. **Maior investimento em testes automatizados deve ser considerado em projetos de Cubesats para viabilizar a execução exaustiva de teste com orçamento reduzido, equipes pequenas e curo prazo**;
- Equipes bem integradas deve ser estratégia tanto em projetos de satélites de pequeno porte quanto de maior porte. Ocorre que em satélites de maior porte a fase integração de sistema requer o envolvimento de um maior número de desenvolvedores e técnicos da qualidade. Em Cubesats os desenvolvedores assumem também o papel dos técnicos da qualidade. **Equipes pequenas não justificam eliminar atividades da garantia da qualidade em processos de desenvolvimento de software embarcado em Cubesat**;
- Ferramentas de gestão de requisitos, rastreabilidade e de testes ajudam as pequenas equipes de desenvolvedores sistematizarem parte das atividades de garantia do produto o que permite evidenciar não conformidades ao longo do processo de desenvolvimento de software. **A frequente mudança de requisitos**

em projetos Cubesat demanda apoio ferramental de gestão de requisitos, rastreabilidade e acompanhamento dos testes de regressão.

8. CONCLUSÃO

Essa dissertação apresentou uma análise do processo de desenvolvimento de software embarcado em subsistema OBC de dois satélites distintos. A análise teve como objetivo encontrar uma resposta à questão de garantir a qualidade do software embarcado em missões baseadas em pequenos satélites, que são severamente limitados em orçamento e prazos, e realizados por pequenas equipes.

O estudo realizado considerou como oportunidade o fato de um software OBDH do satélite de médio porte, no caso o Amazonia-1, ter sido objeto de retrabalho devido a mudanças de requisitos decorrentes da troca de equipamentos em outros subsistemas. Tais mudanças ocorreram nas fases de AIT dos modelos de qualificação e de voo do satélite, o que viabilizou o estudo comparativo com o software OBDH embarcado no NanosatC-BR2. Requisitos voláteis e definições tardias do Sistema e, portanto, chegada tardia de requisitos, são típicas em projetos Cubesat em razão de equipamentos e/ou documentação indisponíveis na fase de desenvolvimento.

Com foco nas questões da garantia da qualidade, como apoio ao processo de desenvolvimento do software, o estudo abordou esforço de verificação e validação com o uso de ferramentas automatizadas de testes, revisões sistemáticas de artefatos de entrega, e adesão aos padrões de qualidade preconizados na Cooperação Europeia para Normalização Espacial (ECSS) ECSS-Q-ST-80C (2009). O uso dos padrões ECSS demonstrou os esforços de ambos satélites estudados para garantir a qualidade e a conformidade dos produtos desenvolvidos, porém buscando reduzir documentação para cumprir prazos.

A pesquisa destaca a importância da adoção de uma ferramenta de gestão de requisitos, casos de teste e rastreabilidade em projetos de software, visando maior precisão nas implementações e modificações. Foram analisadas algumas opções, destacando-se a necessidade de escolha criteriosa considerando o custo da ferramenta e os recursos disponíveis. Durante esse trabalho de pesquisa foi desenvolvido um guia para uso do JAMA, o qual encontra-se no Apêndice A.

Os resultados desta dissertação não estão limitados a uma comparação dos processos adotados no desenvolvimento dos dois satélites estudados e à identificação da necessidade de apoio de uma ferramenta de gestão de requisitos, rastreabilidade e testes,

mas contemplam correlações de características típicas de uma missão Cubesat, tais como baixo custo, curto prazo e esforço em teste de sistema, com especificidades e práticas adotadas nos processos de software embarcado analisados, na perspectiva de garantia da qualidade. As recomendações contidas nesta dissertação visam o desenvolvimento de software de bordo para satélites por pequenas equipes, buscando um equilíbrio entre a qualidade do produto e o esforço necessário para alcançar tal qualidade.

Trabalhos futuros

Sugere-se a criação de um processo de desenvolvimento de software robusto e flexível, que possa ser implementado com sucesso por equipes de diferentes níveis de experiência e que possa suportar a volatilidade dos requisitos e as mudanças inerentes ao desenvolvimento de satélites de pequeno e médio porte.

Com relação às comparações realizadas nessa dissertação, sugere-se aprofundamento em:

1) Avaliar a Maturidade dos processos usando parâmetros com relação a maturidade da equipe e da organização identificando como isso o impacto desses no desenvolvimento do software e a qualidade do resultado final;

2) elaborar diretrizes para um processo de desenvolvimento de software eficaz, om base na análise comparativa e na avaliação de maturidade, considerando a necessidade de projetos de satélites de diferentes portes e complexidades;

3) Testar as diretrizes propostas em um projeto piloto para validar sua eficácia e ajustar conforme necessário, com foco em adaptar o processo para equipes menos experientes e com menos recursos;

4) Determinar como ferramentas para a gestão de requisitos, rastreabilidade, e suporte ao processo de desenvolvimento poderiam ser utilizadas para reduzir a volatilidade dos requisitos e melhorar a eficácia do processo;

5) Estabelecer um processo de feedback e melhoria contínua para garantir que o processo proposto seja adaptável e possa evoluir para atender às necessidades de diferentes tipos de projetos espaciais.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, I. S.; PERONDI, L. F. Gestão da configuração e o ciclo de vida de um projeto na área espacial. In: WORKSHOP DE ENGENHARIA E TECNOLOGIA ESPACIAIS, 1., 2010. São José dos Campos. **Anais...** São José dos Campos: INPE, 2010.

ALMEIDA, D.; MATTIELLO-FRANCISCO, M. F. Modeling of the interoperability between on-board computer and payloads of the nanosat-br2 with support of the uppaal tool. In: IAA LATIN AMERICAN SYMPOSIUM ON SMALL SATELLITES, 1., 2017. **Proceedings…** 2017.

AMBROSIO, A. M. **COFI: uma abordagem combinando teste de conformidade e injeção de falhas para validação de software em aplicações espaciais**. Tese (Doutorado em Engenharia e Tecnologia Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2005.

AMBROSIO, A. M. et al. A conformance testing process for space applications software services. **Journal of Aerospace Computing, Information, and Communication**, v. 3, n. 4, p.146-158, 2006.

AMBROSIO, A. M. et al. Designing fault injection experiments using state-based model to test a space software. In: LATIN-AMERICAN SYMPOSIUM ON DEPENDABLE COMPUTING, 3., 2007. **Proceedings…** Berlin: Springer, 2007. p. 170-178.

ATLASSIAN SOFTWARE. **JIRA.** Disponível em: https://www.atlassian.com/br/software/jira.

BARTIÉ, A. **Garantia da qualidade de software**. [S.l.]: Elsevier, 2002.

BATISTA, C. L. G. et al. Impacts of the space technology evolution in the v&v of embedded software-intensive systems. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND COMPUTATIONAL INTELLIGENCE, 2020. **Proceedings…** 2020. p. 1749-1755.

BATISTA, C. L. G. et al. Towards increasing nanosatellite subsystem robustness. **Acta Astronautica**, v.156, p. 187-196, 2019. Disponível em: https://doi.org/10.1016/j.actaastro.2018.

BRASIL. MINISTÉRIO DE CIÊNCIA, TECNOLOGIA E INOVAÇÃO. **Observatório de tecnologias espaciais**. Brasília: MCTI, 2017.

CONCEIÇÃO, C. A.; MATTIELLO-FRANCISCO, F.; BATISTA, C. C. L. Dependability verification of nanosatellite embedded software supported by a reusable test system. In: LATIN-AMERICAN SYMPOSIUM ON DEPENDABLE COMPUTING, 7., 2016. **Proceedings…** IEEE, 2016. p. 157–163.

CONCEIÇÃO, C. A. **Abordagem sistemática de testes de software comunicante embarcado em nanosatelites com foco em falhas de interoperabilidade**. Dissertação (Mestrado em Engenharia e Tecnologia Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.

DA SILVA, P. D. B.; AMBROSIO, A. M.; VILLANI, E. Model-based testing applied to software components of satellite simulators. **Modelling and Simulation in Engineering**, 2018. Disponível em: https://doi.org/10.1155/2018/3847843.

DINIZ, G. H.; AMBROSIO, A. M.; LAHOZ, C. H. N. Critical software processes tailoring and very small entities (vse): a literature review. **Inernational Journal of Advanced Engineering Research and Science**, v. 6, n. 11, 2019.

EICKHOFF, J. **Simulating spacecraft systems**. New York: Springer, 2009.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-HB-40A: software engineering handbook**. ECSS, 2013.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-ST-40C: software engineering handbook**. ECSS, 2009.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-E-TM-10-21A: space engineering system modelling and simulation**. ECSS, 2010.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-M-ST-10C: space project management: project planning and implementation**. ECSS, 2009.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **ECSS-Q-ST-80C: space product assurance: software product assurance**. ECSS, 2009.

EUROPEAN SPACE AGENCY (ESA). **Demonstration CubeSat projects**. 2016. Disponível em: https://copernicus-masters.com/wp-content/uploads/2017/03/IOD_CubeSat_ECSS_Eng_Tailoring_Iss1_Rev3.pdf.

FISHMAN, C. **They write the right stuff as the 120-ton space shuttle sits surrounded by almost 4 million pounds of rocket fuel, exhaling noxious fumes, visibly impatient to defy gravity, its on-board computers take command**. Disponível em https://www.fastcompany.com/28121/they-write-right-stuff,2013.

FORTESCUE, P.; STARK, J.; SWINERD, G. **Spacecraft systems engineering**. New York: Wiley, 2003. Disponível em: http://onlinelibrary.wiley.com/book/10.1002/9781119971009.

GONÇALVES, R.Q. **Ensino de gerenciamento de projetos de software mediado por ferramentas**. Florianópolis, SC: [s.n.], 2017.

GONZÁLEZ, F. A. D.; CABRERA, P. R. P.; CALDERÓN, C. M. H. Design of a nanosatellite ground monitoring and control software–a case study. **Journal of Aerospace Technology and Management**, v. 8, n. 2, p. 211–231, 2016.

GUTIERREZ, T. et al. Systematic fuzz testing techniques on a nanosatellite flight software for agile mission development source. **IEEE Access**, v. 9, p.114008-114021, 2021.

HIRATA, C.; AMBROSIO, A.M. Combining STPA with CoFI to generate requirements and test cases for safety-critical system. **IEEE Systems Journal**, v. 16, n. 4, p. 6635-6646, 2022.

HASSAN, R. et al. **Toward model-based requirement engineering tool support**. Grand Forks: University of North Dakota, 2017.

IBM. **Engineering requirements management doors**. 2019. Disponível em: https://www.ibm.com/docs/pt-br/ermd/9.6.1?topic=overview-rational-doors.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **3º Aniversário de lançamento do satélite Amazonia-1**. Disponível em: https://www.gov.br/inpe/pt-br/assuntos/ultimas-noticias/3o-aniversario-de-lancamento-do-satelite-amazonia-1.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Processo de desenvolvimento de software de bordo**. São José dos Campos: DIEEC; INPE, 2016.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Relatório de análises post-mortem do processo de desenvolvimento do complemento ao software aplicativo do OBDH o Amazonia-1**. São José dos Campos: DIEEC; INPE, 2021.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Programa de desenvolvimento de CubeSats; NanosatC-BR**. 2024. Disponível em: http://www.inpe.br/sul/nanosat/.

JAMA SOFTWARE. **JAMA connect.** Disponível em: https://requirements.com/Directory/jama-connect. Acesso em: 2023.

JAMA SOFTWARE. **The strategic transition: from word and excel to modern requirements management.** 2023. Disponível em: https://www.jamasoftware.com/ebook/the-strategic-transition-from-word-and-excel-to-modern-requirements-management.

KRIEDTE, W. A new approach to european space standards. **ESA Bulletin**, n. 81, 1995.

KUCINSKIS, F. N. Seminário de introdução à supervisão de bordo de satélites. In: CUBE DESIGN, 2022. **Anais...** São José dos Campos: INPE, 2022.

LANGER, M.; BOUMEESTER, J. Reliability of cubesats: statistical data, developers' beliefs, and the way forward. In: AIAA/USU CONFERENCE ON SMALL SATELLITES, 30., 2016. **Proceedings…** Logan: AIAA; 2016.

LIMA, M. et al. Análise qualitativa do processo de desenvolvimento de software embarcado no Satélite Amazonia-1. In: WORKSHOP EM ENGENHARIA E TECNOLOGIA ESPACIAIS, 13., 2022, São José dos Campos. **Anais...** São José dos Campos: INPE, 2022.

MATTIELLO-FRANCISCO, M. F. et al. Inrob: an approach for testing interoperability and robustness of real-time embedded software. **Journal of Systems and Software**, v. 85, n. 1, p. 3– 15, 2012.

MATTIELLO-FRANCISCO, M. F. et al. A brazilian software industry experience in using ecss for space application software development. In: LOUREIRO, G.; CURRAN, R. (Ed.). **Complex systems concurrent engineering**. Berlin: Springer, 2007. p.167–174.

MIRANDA, D. J. F. et al. A comparative survey on flight software frameworks for 'new space' nanosatellite missions. **Journal of Aerospace Technology Management,** v. 11, 2019.

NATIONAL AERNAUTICS AND SPACE ADMINISTRATION (NASA). **NASA-STD-8739.8B: software assurance and software safety standard**. Disponível em: https://standards.nasa.gov/sites/default/files/standards/NASA/B/0/NASA-STD-87398-RevisionB.pdf.

PINHEIRO, A. C.; SIMÃO, A.; AMBROSIO, A. M. FSM-based test case generation methods applied to test the communication software on board the itasat university satellite: a case study. **Journal of Aerospace Technology Management**, v.16, n.4, 2014.

PINTO, A. C. **O que aprendemos com a NASA e a SpaceX sobre o preditivo e o ágil**. 2020. Disponível em: https://www.linkedin.com/pulse/o-que-aprendemos-com-nasa-e-spacex-sobre-preditivo/?trackingId=wrX08PeKTfyQtWjc2aGa7Q%3D%3D.

PONTES, R. P. et al. Embedded critical software testing for aerospace applications based on PUS. In*:* WORKSHOP DE TESTES E TOLERÂNCIA A FALHAS, 11., 2010, Gramado/RS. **Anais...** Porto Alegre: SBC, 2010. p. 119-132.

PONTES, R. P. et al. Contributions of model checking and CoFI methodology to the development of space embedded software. **Empirical Software Engineering**, v.19, p. 39–68, 2014. Disponível em: https://doi.org/10.1007/s10664-012-9215-y.

ROHLING A. J. **A reference architecture for satellite systems operations**. Tese (Doutorado em Engenharia e Tecnologia Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2018.

SCHUCH, N. J. et al. NANOSATC-BR2 launch: the NANOSATC-BR Cubesat developement program status & future. In: INTERNATIONAL ACADEMY OF ASTRONAUTICS LATIN AMERICAN CUBESAT WORKSHOP, 4., 2020, São José dos Campos, SP. **Proceedings…** 2020.

SOMMERVILLE, I. **Software engineering**. 9.ed. [S.l.]: Pearson Education, 2014.

SWEETING, M. N. Modern small satellites - changing the economics of space. **Proceedings of the IEEE**, v. 106, n. 3, 2018.

WAYRONE, K. L. S.; GRANDE, E. T. G.; OLIVEIRA, D. C. Estudo do satélite brasileiro Amazonia-1 e de sua trajetória: mapeamento sistemático e análise documental dos artefatos históricos oficiais. **Research, Society and Development**, v. 11, n. 2, e29011225894, 2022.

VILLELA, T. et al. Towards the thousandth CubeSat: a statistical overview. **International Journal of Aerospace Engineering**, v, 2019, e5063145, 2019.

ZAIDI, Y. et al. Rapid, automated, test, verification and validation for the CubeSats. **International Journal of Space Science and Engineering**, v. 5, n. 3, p. 242-268, 2019.

APÊNDICE A - GUIA PARA UTILIZAÇÃO DA FERRAMENTA JAMA NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Durante esse trabalho de pesquisa foi desenvolvido um guia para uso do JAMA. O guia contempla 6 passos, como segue:

(1) Planejamento: Definir estratégias para a incorporação da ferramenta, identificando as equipes-chave e estabelecendo metas claras para a transição.

(2) Levantamento de Requisitos: Utilizar as capacidades da ferramenta para centralizar e organizar os requisitos, facilitando a colaboração e a rastreabilidade.

(3) Design: Integrar a ferramenta no processo de design, promovendo uma visão holística da arquitetura do software e otimizando a comunicação entre as equipes.

(4) Implementação: Utilizar a ferramenta para documentar e rastrear o progresso da implementação, promovendo uma abordagem ágil e adaptativa.

(5) Testes: Incorporar a ferramenta nos processos de teste, facilitando a organização de casos de teste, registros de resultados e a análise de cobertura.

(6) Implantação: Documentar a versão final do software da ferramenta, garantindo a integridade da informação para futuras referências e auditorias.

APÊNDICE B - ALGUMAS FERRAMENTAS DE GESTÃO DE REQUISITOS

A escolha de uma ferramenta para guiar o desenvolvimento de software em um ambiente complexo é crucial e impacta diretamente na eficiência operacional, no cumprimento de prazos e na contenção de despesas em um cenário de orçamento restrito. Embora a preferência inicial tenha sido a Jama, dada a sua suposta adequação a projetos complexos, a ponderação dessa escolha à luz do custo associado trouxe à tona uma realidade onde a viabilidade econômica em projetos com recursos financeiros limitados emerge como um fator decisivo na escolha da ferramenta de gerenciamento de requisitos. Adoção de uma ferramenta de gestão de requisitos, com rastreabilidade a casos de teste e resultados de teste, compatível com o orçamento do projeto permitiu mais precisão nas modificações, manutenção do histórico das modificações com baixo custo, uma vez que não seria necessária mão-de-obra extra para o controle de versões e rastreabilidade nas modificações.

A Tabela B.1 sumariza a avaliação das ferramentas.

Tabela B.1 - Algumas Ferramentas de Gerenciamento de Requisitos.

Ferramentas	Jama	Jira	IBM Rational Doors
Rastreabilidade e Análise de Impacto	x	x	x
Gerenciamento de Teste	x	x	x
Conexão em tempo real durante os testes, mesmo estando em outro departamento	x	x	x
Sistema de Aprovação	x	x	x
Roteiros (ou caso de testes)	x	x	x
Link de Itens afetados (rastreável)	x	x	x
Lista de Pendências	x	x	x
Sistema de Aprovação de Roteiros	x	x	x
Gerador de Relatórios	x	x	x
É possível integrar com outras ferramentas do Rational Incluindo o Rational Team Concert, Rational Quality Manager, Rational Doors Next Generation, entre outros relacionados ao programa Rational Doors.	x	x	x
Armazenamento e Gerenciamento de Requisitos	x	x	x
Plataforma Web	x	x	x
Controle de Acesso	x	x	x
Gerenciamento de Mudança	x	x	x
Acompanhamento do Status do Requisito	x	x	x
Rastreabilidade de Requisitos	x	x	x
Importar/Exportar Conteúdo	x	x	x
Recurso de Colaboração	x	x	x
Integração com Outras Ferramentas como Excel, Doors, Jira. Trelo entre outros.	x	x	x

Fonte: JAMA, JIRA e IBM (2024).

A análise de algumas ferramentas ressaltou a importância da escolha criteriosa de ferramentas de gerenciamento de requisitos e a relação custo versus disponibilidade de recursos. Outros aspectos cruciais incluem integração, adaptação a metodologias ágeis e recursos de relatórios personalizáveis. A escolha certa de uma ferramenta de requisitos não apenas facilita o gerenciamento de requisitos, mas também contribui para o sucesso das missões uma vez que melhora sua eficiência e possibilita uma colaboração mais efetiva entre as equipes.

Para o caso de aderir a uma ferramenta de requisitos, uma outra sugestão é a adoção dos registros da ferramenta de gestão de requisitos como entregáveis da missão, evitando a reemissão de informações como documentos, e o consequente esforço de adequação e formatação, que pode ser considerável, a depender da quantidade de requisitos e testes. Isso envolveria um acordo com a autoridade de Engenharia de Qualidade, ou aquele que cumprir seu papel (muito provavelmente o coordenador ou engenheiro principal do projeto).

O uso do JAMA seguiu um procedimento orientador elaborado pelo Grupo de Supervisão de Bordo, indicando que a ferramenta é um Sistema de Gerenciamento de Requisitos online, compartilhado entre os envolvidos no desenvolvimento. Ele registra requisitos, atividades relacionadas à sua evolução e modificações, com alertas sobre impactos em requisitos relacionados. Além disso, o sistema mantém a rastreabilidade entre requisitos e testes, registrando diretamente os resultados dos testes.

O procedimento dito acima, também destaca a conformidade do JAMA com as expectativas do Handbook ECSS-E-HB-40A (2012), mencionando capacidades como garantia de identificadores únicos, anexação de atributos aos requisitos, suporte à rastreabilidade, controle de versões e histórico de revisões, acesso concorrente multiusuário e interface com ferramentas de documentação.

As ferramentas alternativas ao JAMA, com funcionalidades similares, porém mais acessíveis a projetos menores são recomendadas. O Handbook ECSS-E-HB-40A (2012) indica que ferramentas desse tipo vêm sendo utilizadas na área espacial desde o início dos anos 2000. O Handbook também aponta as seguintes capacidades esperadas para uma ferramenta como essa:

Aspectos essenciais incluem garantia de identificadores únicos, anexação de atributos aos

requisitos (como status, importância, versão, método de verificação, criador, data de criação, última data de modificação, risco, categoria etc.), capacidade de captura, modificação, deleção e pesquisa de requisitos e seus atributos. Adicionalmente, são considerados requisitos como suporte à referência cruzada e rastreabilidade, controle de versões e histórico de revisões, acesso concorrente multiusuário com eficiente controle de acesso, interface com ferramentas de documentação para facilitar a geração de documentos e a presença de recursos de visualização e multimídia.

Essas capacidades, derivadas do Handbook ECSS-E-HB-40A (2012), fundamentam a escolha de ferramentas que proporcionam ambientes robustos e colaborativos para o desenvolvimento de software em projetos de mestrado.

Com base nos resultados obtidos, recomenda-se para novos projetos de cubesats, o uso de uma ferramenta de gerenciamento de requisitos adequada a projetos com orçamento restrito, que haja um balanço eficaz da ferramenta com a realidade financeira do projeto.

A análise da facilidade de uso, integração com outras ferramentas e suporte técnico proporcionou *insights* para a eficácia operacional. Nesse contexto, o Jama não apenas apresentou uma alternativa mais acessível, mas manteve funcionalidades essenciais, garantindo uma solução equilibrada e pragmática para os desafios enfrentados em projetos espaciais.

O desenvolvimento do software de supervisão de bordo para o satélite Amazonia-1 foi um processo desafiador, mas bem-sucedido. O processo seguiu as recomendações da ECSS e adotou uma abordagem iterativa, permitindo modificações de requisitos entre as etapas e identificação de novas versões. Apesar de não estabelecer uma relação direta entre as versões incrementais do software e os marcos da missão, o processo foi adaptado às necessidades do projeto.

ANEXO A – CONTRIBUIÇÕES DA FERRAMENTA JAMA PARA O PROJETO DO SOFTWARE DO AMAZONIA-1

A seguir, destacam-se as contribuições específicas da ferramenta JAMA em cada fase do ciclo de vida realizadas durante o desenvolvimento do Projeto do software do OBDH do satélite AMAZONIA-1 (INPE, 2016).

CICLO DE VIDA ECSS	ATIVIDADES
Composição do baseline de requisitos técnicos **Fase A da norma ECSS-M-ST-10C**	Nesta fase acontece o processo que estabelece o baseline de requisitos técnicos para o OBDH, onde é planejado todo o ciclo de desenvolvimento do s*oftware* até a entrega final que é a sua última versão, nesta fase se estabelece também os marcos de entregas seguindo o cronograma do subsistema do satélite bem como as restrições de projeto. Aqui é extraído o documento *Software* development plan (SDP).
Atividades da fase versus Jama	Nesta primeira fase que inicia a contribuição do programa Jama no desenvolvimento do Software do OBDH, onde deve extrair todos os requisitos possíveis do "cliente" e anotá-los, fazendo a compilação e grau de importância. Isso por que tem um módulo no Jama onde os requisitos são anotados e armazenados de forma segura, foi desses requisitos organizados dentro do sistema que surgem os documentos de Requisitos de Software exigido pela Norma ECSS, como o *Software System specification (SSS), Software Requirements Specification (SRS) e o Software Review plan (SRevP*) – complementar.
Instrução Jama	O primeiro nível de requisitos inseridos no Jama deve ser o de requisitos técnicos de mais alto nível do projeto. Os requisitos do cliente devem ser referenciados textualmente por seus identificadores, através de campo específico para tal fim, no formulário de requisitos. Isso tem por objetivo reduzir a complexidade da rastreabilidade dos requisitos, uma vez que o baseline para o processo será composto pelos requisitos técnicos.
Revisão dos requisitos do Subsistemas (SRR) **Fase B da norma ECSS-M-ST-10C**	Na composição do baseline dos requisitos técnicos pode-se obter como saída os requisitos em alto nível que serão apresentadas para o cliente e fornecedor na SRR (Revisão dos Requisitos do Subsistema) Neste momento será feito um acordo formal sobre o entendimento dos requisitos necessários para o projeto em questão, juntamente com a entrega dos documentos que a norma ECSS sugere o *Software System Specification (SSS), Software Requirements Specification (SRS) e o Software Review plan (SRevP)* – complementar (INPE, 2016). Aqui se encerra a etapa de Revisão de Planejamento do Desenvolvimento de Software, na qual o plano foi submetido à mesma banca da SRR
Instrução Jama	Revisões de requisitos, de testes e processos de análise de impactos devem ser suportadas pelo recurso 'Reviews' do Jama, que permite a revisão on-line e assíncrona de itens do banco de dados por usuários do sistema e também convidados. O Gerente de Requisitos deve assumir o papel de moderador da revisão, gerenciando o progresso dos revisores e zelando pelo cumprimento do prazo estabelecido para o término da revisão. Idealmente, revisões presenciais devem ser realizadas somente após o término das revisões on-line, focando majoritariamente nos pontos de discordância entre os revisores.

CICLO DE VIDA ECSS	ATIVIDADES
Fases de Preparação da verificação do produto	"Essa etapa, poderá ocorrer em paralelo à definição da arquitetura do produto (Fase B de acordo com o ECSS-M-ST-10C), visa o refinamento dos requisitos, através de sua decomposição e alocação a elementos de mais baixo nível da arquitetura do produto, bem como a definição de como o atendimento aos requisitos será verificado") (INPE, 2016). É nesta etapa que se constitui as alocações dos requisites técnicos para cada elemento da arquitetura definida para o produto até o menor nível verificável.
Atividades da fase versus Jama	O Jama nesta etapa contribui da disposição das informações de requisitos e nas atualizações dos requisitos que surgem durante o processo de verificação das interfaces. Nesta etapa surgem os documentos de *Software design document (SDD) e Software release document (SDD*) (INPE, 2016).
Revisão de projeto preliminar (PDR) **Fase B da norma ECSS-M-ST-10C**	"No que diz respeito ao presente processo, a PDR tem por objetivo principal a verificação do projeto preliminar do produto contra os requisitos." Todas as saídas da etapa anterior são objetos de revisão e, portanto, entradas para a PDR. A saída da etapa é a aprovação formal do cliente quanto ao projeto, planos e demais documentos, após o atendimento dos eventuais itens de ação gerados quando da PDR (INPE, 2016).
Fase do Projeto inicial do software (Etapa)	Nessa etapa que se inicia o desenvolvimento do software, isto é, a codificação que resulta nos comandos do OBDH. Deverá ser realizado conforme os requisitos entendidos pela equipe. Desta forma acontece o projeto inicial utilizando os documentos entregue da etapa anterior. Todo o desenvolvimento leva em conta os documentos do sistema ICD (*Interface control document*) que é a base do projeto, esta etapa se encerra com a revisão de projeto preliminar (PDR) (INPE, 2016).
Atividades da fase versus Jama	O Jama nesta fase contribui no armazenamento e organização das informações de requisitos e atualizações que surgem durante o processo de desevolvimento. Em paralelo serve como base de informações para que ocorrece também o plano e os procedimentos de testes do software em desenvolvimento. No programa Jama podem ser descritos todo o processo de caso de testes, sendo uma base para a realização dos testes no LIT ou Laboratório de Integração. Os testes podem ser realizados em um computador integrado nos simuladores dos equipamentos, que simularam as interfaces do OBDH. O Sistema Jama é quem gera os relatórios de resultados dos testes executados e é possível que os responsáveis possam acompanhar em tempo real de qualquer local, bastando ter no seu computador o programa Jama instalado. Nessa fase é elaborado os documentos de *Software Verification Plan (SVerP), Software Validation Plan (SVaip), Software Unit/Integration Test Plan (SUITP), Software Design Document (SDD) e Software Release Document (SDD)* (INPE, 2016).
Desenvolvimento de Cada Versão do Software	Aqui se inicia o processo de desenvolvimento das versões incrementais do software. Nesta fase são realizadas as reuniões técnicas para cada entrega a cada reunião identifica os novos requisitos a cada novos requisitos devem ser definidos as atividades, tais como: novos Casos de Testes, Codificação, Testes Unitários, Revisão 'Por Pares' do Código, Composição dos Procedimentos de Testes, Preparação dos Recursos Para-Testes, Emissão de Versão para Testes, Campanha de Testes, Revisão Técnica de Entrega da Versão e finalmente a Configuração dos Produtos de cada nova Versão (INPE, 2016).

CICLO DE VIDA ECSS	ATIVIDADES
Atividades da fase versus Jama	O Jama nessa fase contribui na disposição das informações de requisitos e atualizações de requisitos que surge durante o processo de verificação das interfaces, nos casos de testes, nos testes de "pares," na emissão de novas versões de procedimentos de testes e na documentação desta etapa. Nesta etapa surgem os documentos *Software Interface Document (IRD) e revisão do Software System Specification (SSS) e Software Released (SReID*).
Instrução Jama	Requisitos provenientes de especificações gerais de programa, como o 'Design and Construction' e 'Environmental Specification' devem constar de projetos específicos no Jama (um projeto por documento), sendo compartilhados para o projeto específico do produto a ser desenvolvido através do recurso 'reuse items.'
Fase do detalhamento da verificação do produto	Nesta etapa já está caminhando para a fase C de acordo com a ECSS-M-ST-10C, onde são detalhados toda a verificação do produto por meio dos procedimentos de testes e composição dos relatórios de verificação do projeto (INPE, 2016).
Atividades da fase versus Jama	O Jama nesta etapa contribui na disposição das informações de requisitos e atualizações de requisitos que surge durante o processo verificação das interfaces, nos casos de testes, nos testes de "pares," na emissão de novas versões de procedimentos de testes e na documentação desta etapa. Nesta fase surgem as revisões detalhadas dos documentos de *Software Interface Document (IRD), Software system Specification (SSS), Software Release (SReID) e Software Verification Report (SVR).*
Instrução Jama	Em caso de desvio de um requisito reutilizado de um projeto de especificações gerais (como o 'Design and Construction'), o vínculo do requisito específico com o projeto original deve ser 'quebrado' através do comando '*Break Sync'* do Jama. Tal ação deve ser justificada no histórico do requisito, fazendo referência ao documento configurado que no embasa.
Revisão de projeto detalhado (CDR) **Fase C da norma ECSS-M-ST-10C**	Na fase CDR deve ser realizada a avaliação do projeto final do produto e o status de qualificação juntamente com os stakeholders (INPE, 2016).
Atividades da fase versus Jama	O Jama nesta fase contribui na disposição das informações de requisitos e atualizações de requisitos que surge durante o processo verificação das interfaces, nos casos de testes, nos testes de "pares," na emissão de novas versões de procedimentos de testes e na documentação desta etapa. Nesta fase deverão ser entregues os documentos de *Software Interface Document (IRD), Software System Specification (SSS), Software Released (SReID) e Software Verification Report (SVR).*
Instrução Jama	O Gerente de Requisitos deve verificar a completeza da rastreabilidade através de recursos como 'Trace Matrix' e 'Coverage Explorer' do Jama.
Verificação do produto **Fase D da norma ECSS-M-ST-10C**	Nesta etapa do processo são realizados os testes de verificação juntamente com a qualificação do produto onde se verifica se o produto atende todos os requisitos do projeto (INPE, 2016).
Atividades da fase versus Jama	O Jama nesta fase contribui na disposição das informações de requisitos e atualizações de requisitos que sugem durante o processo verificação das interfaces, nos casos de testes, nos testes de "pares," na emissão de novas versões de procedimentos de testes e na documentação desta etapa, bem como a realização dos testes de verificação onde foram gerados os relatórios automáticos pelo Jama. Nesta fase devem ser gerados e revisados os documentos de *Software User Manual (SUM), Software Verification Reposta (SVR), Software Design Document (SDD), Software System Specification (SSS), e Software Released (SReID).*

CICLO DE VIDA ECSS	ATIVIDADES
Instrução Jama	Problemas encontrados durante testes devem ser reportados através de '*defects*' do Jama, e devidamente relacionados ao teste no qual o problema foi identificado.
Revisão de qualificação (QR) e de aceitação (AR) **Fase D e da norma ECSS-M-ST-10C**	Nesta fase são verificados junto aos stakeholders se o produto atende todos os requisitos do projeto obtendo assim a aceitação final. Nesta fase devem ser entregues os documentos de *Software User Manual (SUM), Software verification reposta (SVR), Software Design Document (SDD), Software System Specification (SSS), e Software Released (SReID)* (INPE, 2016).
Fase de Configuração dos Produto da Versão	Nesta etapa éconfigurado adequadamente todos os documentos do projeto ou missão da qual o SW do OBDH gerou durante o processo de desenvolvimento, incluindo os produtos da iteração do ciclo de desenvolvimento de software, incluindo os elementos necessários à reprodução dos testes (como bancos de dados de TC e TM) e as evidências de testes (como logs de testes e registros de comunicação entre equipamentos) (INPE, 2016).
Instrução Jama	As 'tags' a serem utilizadas no projeto devem ser padronizadas no escopo do projeto, evitando-se que cada usuário crie seu próprio conjunto de tags, o que minimizaria a efetividade da ferramenta devido a um grande número delas.
Revisão de Entrega da Versão Final do OBSW	Nesta etapa são analisados todos os testes de verificação e validação dos produtos, testes estes que foram realizados no satélite FM, no AIT do LIT com a versão final do software. A revisão técnica tem por objetivo de submeter a banca técnica o resultado do processo de desenvolvimento da versão do SW do OBDH para sua devida aceitação (INPE, 2016). Nesta etapa devem ser entregues os documentos de *Software User Manual (SUM), Software Verification Reposta (SVR), Software Design Document (SDD), Software System Specification (SSS), e Software Released (SReID).*
Instrução Jama	Itens de Ação internos do SUBORD (gerados no grupo, e a serem executados pelo grupo), no escopo do desenvolvimento do projeto, devem ser gerenciados no próprio Jama, através do uso de formulários de *'Action Itens'* criados especificamente para esse fim. Isso não se aplica aos Itens de ação dos programas do INPE, gerenciados através do *Windchill*.
Instrução Jama	Todas as discussões, mensagens, registros de justificativas e questionamentos entre membros da equipe, através do Jama, devem ser em português – mesmo que o idioma oficial do programa seja outro. Tais informações são consideradas registros de trabalho em andamento, não sendo configuráveis.
Análise após ecerramento do Processo.	Nesta etapa é realizada uma análise de todo o processo de desenvolvimento do software, com a intenção de encontrar lições aprendidas nos casos de sucesso ou fracasso (INPE, 2016).
Atividades da fase versus Jama	O Jama nesta etapa contribiu na disposição das informações para as inúmeras minutas que foram elaboradas durante o processo que foram a base das informações para as análises após o encerramento do processo.

Legenda de cores do guia do Jama	
Azul	Ciclo de Vida ESCC
Cinza	Atividades no Jama
Branco	Fases intermediárias
Verde	Instrução Jama