XMITS 3.1: APLICAÇÃO DE VERIFICAÇÃO FORMAL COM APOIO COMPUTACIONAL

CAMILA P. SALES[1], LUCIANA B. R. DOS SANTOS[2], LUCAS V. POVOA[3]

[1] Graduanda em Tecnologia em Análise e Desenvolvimento de Sistemas, Bolsista PIBITI, IFSP, Câmpus Caraguatatuba, camilapsales27@gmail.com
[2] Professora IFSP, Orientadora, Doutora em Computação Aplicada, INPE - Instituto Nacional de Pesquisas Espaciais, lurebelo@ifsp.edu.br
[3] Professor IFSP, Doutorando em Engenharia Eletrônica e Computação, ITA - Instituto Tecnológico de Aeronáutica, venezian@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 1.03.01.02-0 Linguagens Formais e Autômatos

RESUMO: Este trabalho tem como objetivo a adição de uma extensão à ferramenta XMITS, que é um conversor de diagramas comportamentais UML em sistemas de transição de estados. A extensão visa mostrar onde as inconsistências, apontadas pelo *Model Checker* utilizado, ocorrem no(s) diagrama (s) que a(s) originou, bem como estimar, por meio de aprendizagem de máquina, a(s) inconsistência(s) encontrada(s) entre o(s) diagrama(s) e o(s) requisito(s). Estas funcionalidades buscam facilitar a verificação de inconsistências entre os diagramas UML e os requisitos do software.
PALAVRAS-CHAVE:Verificação Formal; Diagramas UML; XMITS; Algoritmos de Aprendizagem.

1 INTRODUÇÃO

No âmbito de desenvolvimento de software métodos de V&V (Validação e Verificação) se mostram essenciais para assegurar a alta qualidade das aplicações, especialmente em aplicações críticas. Contudo, não são consideravelmente utilizados, devido demandarem muito tempo para serem aplicados em sistemas de software e hardware de alta complexidade.

Nesse cenário, foi desenvolvida a ferramenta XMITS (XML *Metadata Interchange to Transition System*), que segundo Eras et al. (ERAS,2014) objetiva permitir a conversão de diagramas comportamentais UML em sistemas de transição de estados, visando criar um elo entre diagramas UML (*Unified Modeling Language*), comumente utilizados para documentação de software, e métodos de Verificação Formal, mais especificamente o *Model Checking*.

A versão atual da XMITS trabalha com três diagramas comportamentais: sequência, atividades e máquina de estado. Contudo, para que a XMITS possa ser utilizada no dia a dia dos usuários, algumas deficiências ainda precisam ser sanadas.

O objetivo do presente trabalho é desenvolver uma extensão para a ferramenta XMITS, que mostre onde as inconsistências (contraexemplos) apontadas pelo *Model Checker* ocorrem no(s) diagrama(s) que as originaram.

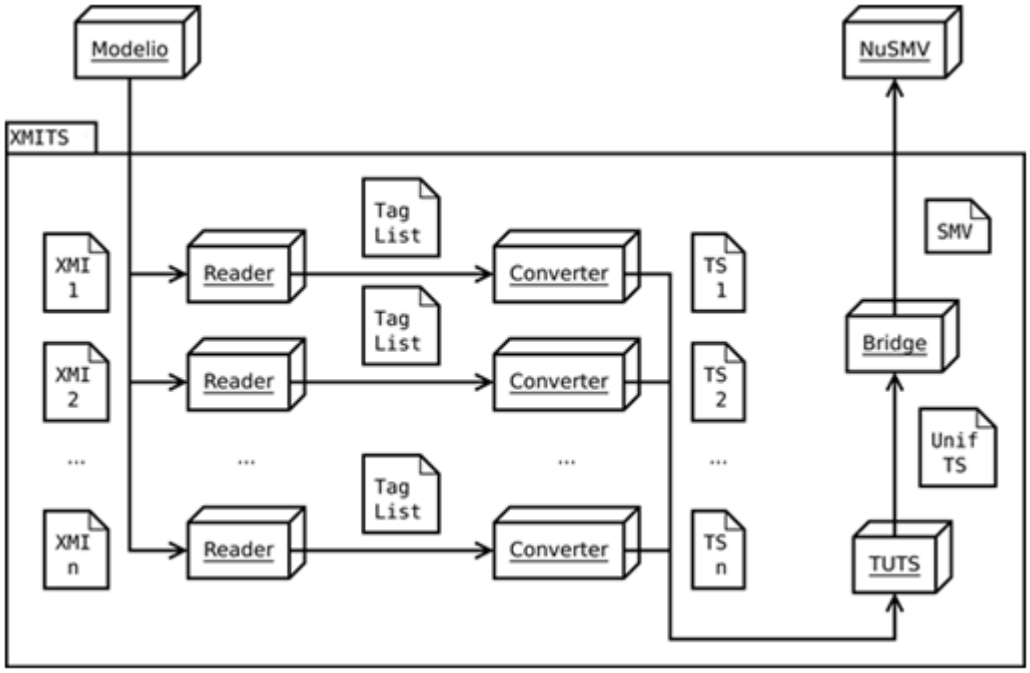
Adicionalmente, para verificar os defeitos encontrados, no que diz respeito às inconsistências entre o sistema (representado pelos diagramas) e os requisitos do software,

utilizou-se *Machine Learning* para que a estimativa da inconsistência pudesse ser realizada de forma automática, baseando-se apenas na base de treinamento utilizada.

2 MATERIAL E MÉTODOS

A ferramenta XMITS foi desenvolvida utilizando-se a linguagem de programação Java, segundo(LEAHY, 2018), conhecida como uma linguagem de alto nível. A ferramenta recebe arquivos em formato XMI, construídos por meio do Modelio, os processa e converte em um Sistema de Transições (TS), base para entrada da ferramenta de *Model Checking* NuSMV, que segundo (FERNANDES, 2011) permite descrever sistemas computacionais síncronos e assíncronos através de um sistema de transição de estados. O fluxo de diagramas na ferramenta pode ser visto na Figura 1.

FIGURA 1. Ilustração do fluxo interno da ferramenta XMITS.



Fonte: SANTOS(2015)

A linguagem Java, foi também utilizada para o desenvolvimento da primeira funcionalidade construída neste projeto, que contém as funções responsáveis pelo tratamento do contraexemplo, ou seja, a identificação dos pontos exatos dentro dos diagramas onde as inconsistências foram originadas.

Para desenvolvimento da segunda parte deste projeto foi utilizada a linguagem de programação Python, segundo (LUKASZEWSKI, 2018) é uma linguagem de programação de uso geral que pode ser usada em qualquer sistema operacional de computador moderno. Ele pode ser usado para processar texto, números, imagens, dados científicos.

Utilizou-se algoritmos de aprendizagem de máquina, que segundo (MATOS, 2015) são conjuntos de instruções e regras que permitem que os computadores possam agir e tomar decisões baseados em dados ao invés de ser explicitamente programados para realizar uma determinada tarefa. Podem ser divididos em aprendizado supervisionado e não supervisionado.

Neste projeto foram utilizados algoritmos de aprendizagem de máquina supervisionados, por meio dos seguintes modelos:

i) modelo de conjunto *Extra Trees Classifier* - ExTree (DA SILVA, 2018) onde todos os nodos escolhem o melhor valor entre todas as características e os pontos possíveis de corte;

ii) modelo linear *Logistic Regression* - LogReg (DA SILVA, 2018), é um algoritmo de classificação, a regressão logística realiza classificação binária, portanto os rótulos de saída são dicotômicos;

iii) rede neural *Multilayer Perceptron* - MLP (KLEIN, 2011 - 2018) é um modelo de rede neural artificial *feedforward* que mapeia conjuntos de dados de entrada em um conjunto de saídas apropriadas. Um MLP consiste em várias camadas de nós em um grafo direcionado, com cada camada totalmente conectada ao próximo. Exceto pelos nós de entrada, cada nó é um neurônio; e

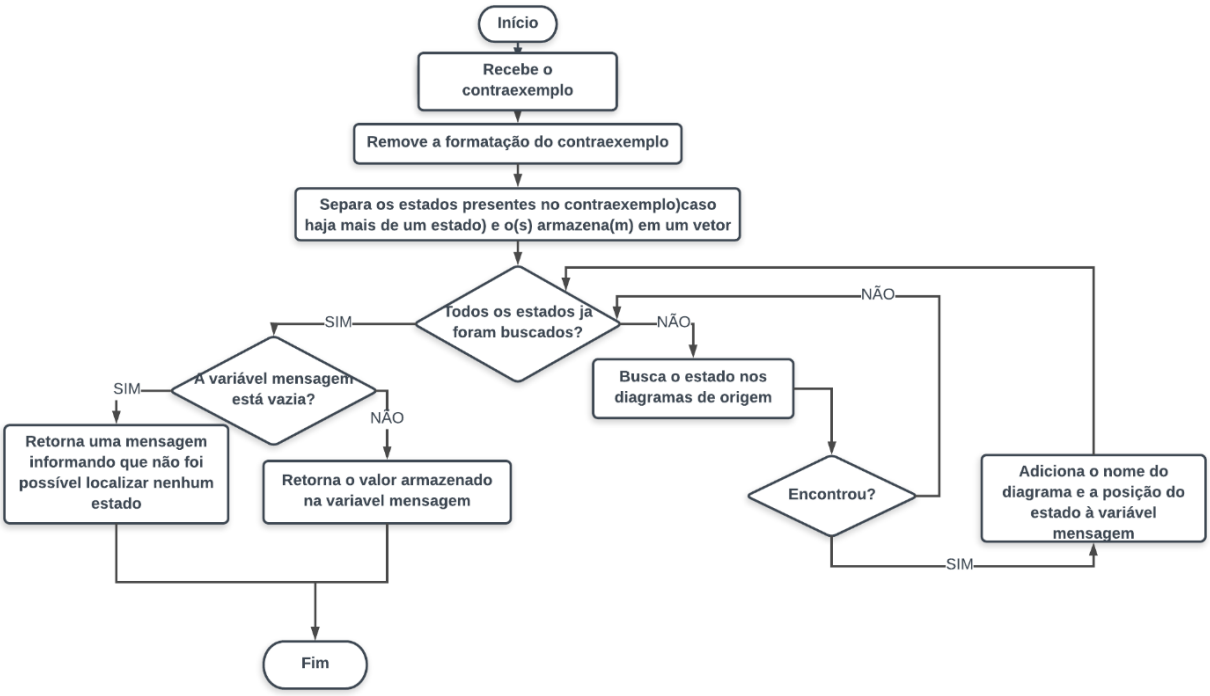
iv) modelo *eXtreme Gradient Boosting* - XGB (NISHIDA, 2017) é uma técnica de aprendizagem de máquina para problemas de regressão e classificação, que produz um modelo de previsão na forma de um conjunto de modelos de previsão fracos, geralmente árvores de decisão.

3 RESULTADOS E DISCUSSÃO

Como resultado deste trabalho foram desenvolvidas duas novas funcionalidades que foram acrescidas à ferramenta XMITS.

A primeira, responsável por relatar ao usuário a origem do(s) estado(s) presente(s) no contraexemplo gerado após a aplicação do Model Checking, tem seu fluxograma mostrado na Figura 2.

FIGURA 2. Fluxograma da primeira funcionalidade.



Fonte:Autor(2018)

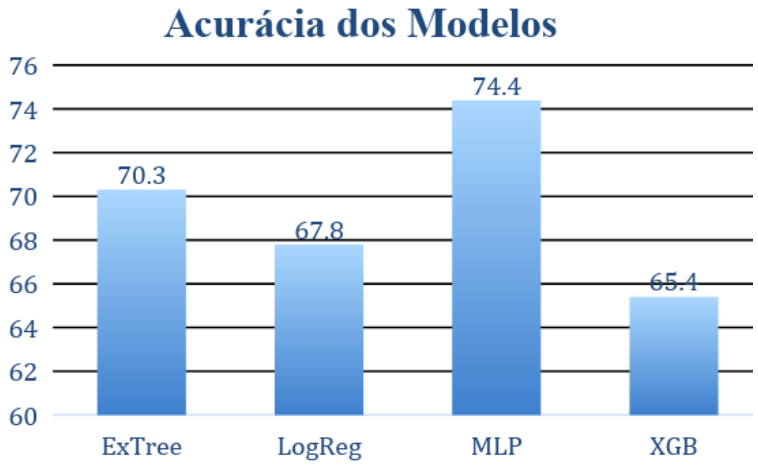
A segunda funcionalidade realiza a estimativa de existência de inconsistências entre os requisitos e os diagramas que os representam, onde utilizou-se aprendizagem de máquina.

A fim de criar uma base de dados para os algoritmos, foram utilizados estudos de casos reais no âmbito espacial. Foi gerada uma tabela com os dados extraídos dos diagramas e

seus respectivos requisitos, onde os resultados são: i) a existência ou não de inconsistência, e caso exista inconsistência, ii) a classificação da inconsistência, de acordo com as classes de erros presentes no estudo.

Os algoritmos estudados obtiveram porcentagem de acurácia aceitáveis para cumprimento do objetivo, uma vez que, o algoritmo com menor acurácia foi XGB, com acurácia de 65.4%, e o com maior o MLP obteve acurácia de 74,5%, sendo este o escolhido para ser aplicado nesta funcionalidade. A acurácia dos modelos pode ser visualizada na Figura 3.

FIGURA 3. Acurácia dos modelos escolhidos para o estudo.



Fonte: Autor(2018)

4 CONSIDERAÇÕES FINAIS

Duas novas funcionalidades foram adicionadas à ferramenta XMITS, auxiliando sua utilização, uma vez que facilitam a verificação das inconsistências.

O objetivo de informar identificar o diagrama de origem do contra exemplo foi devidamente implementado, além de uma funcionalidade que realiza a estimativa de ocorrência de inconsistências. A implementação de um módulo, responsável por retornar ao usuário as classes de erro que o contraexemplo se encaixa, deve ser realizada em trabalhos futuros como trabalhos futuros.

5 AGRADECIMENTOS

Agradecemos ao CNPq, pelo apoio financeiro por meio da bolsa PIBITI e ao IFSP câmpus Caraguatatuba, por conceder a estrutura necessária para realização deste projeto.

REFERÊNCIAS

DA SILVA, J. C. Algoritmos de Aprendizagem de Máquina: qual deles escolher?, 2018. Disponível em: <https://medium.com/machina-sapiens/algoritmos-de-aprendizagem-de-m%C3%A1quina-qual-deles-escolher-67040ad68737>. Acesso em: 15 agosto 2018.

ERAS, E. R. XMITS: Um Leitor e Conversor de Arquivos XML Para Sistema de Transição de Estados, São José dos Campos, 2014.

FERNANDES, F. UM ARCABOUÇO PARA VERIFICAÇÃO AUTOMÁTICA DE MODELOS UML, 2011.

KLEIN, B. D. Neural Networks with scikit, 2011 - 2018. Disponível em: <https://www.python-course.eu/neural_networks_with_scikit.php>. Acesso em: 15 agosto 2018.

LEAHY, P. What Is Java?, 2018. Disponível em: <https://www.thoughtco.com/what-is-java-2034117 >. Acesso em: 23 agosto 2018.

LUKASZEWSKI, A. What Is Python Programming Language?, 2018. Disponível em: <https://www.thoughtco.com/what-is-python-2813564>. Acesso em: 24 agosto 2018.

MATOS, D. Conceitos Fundamentais de Machine Learning, 2017. Disponível em: <http://www.cienciaedados.com/conceitos-fundamentais-de-machine-learning/>. Acesso em: 19 julho 2018.

NISHIDA, K. Introduction to Extreme Gradient Boosting, 2017. Disponível em: <https://blog.exploratory.io/introduction-to-extreme-gradient-boosting-in-exploratory-7bbec554ac7>. Acesso em: 10 agosto 2018.

SANTOS, L. B. R. A Methodology to Apply Formal Verification to UML-Based Software. , São José dos Campos, 2015.