Revista Científica On-line
Tecnologia – Gestão – Humanismo
ISSN: 2238-5819

FERRAMENTAS PARA ANÁLISE DE DEPENDABILIDADE E SEGURANÇA DE SOFTWARES: UM BREVE LEVANTAMENTO

TOOLS FOR DEPENDABILITY ANALYSIS AND SAFETY SOFTWARE: A BRIEF SURVEY

HERRAMIENTAS PARA EL ANÁLISIS DE CONFIABILIDAD Y SEGURIDAD DE SOFTWARE: UNA BREVE ENCUESTA

Glauco da Silva[1,2] (glaucogs@iae.cta.br)
Carlos Henrique Netto Lahoz[1,2] (lahozchnl@iae.cta.br)

[1] IAE - Instituto de Aeronáutica e Espaço
[2] ITA - Instituto Tecnológico de Aeronáutica

Resumo
A utilização de ferramentas para auxílio de análise de dependabilidade e segurança de software pode contribuir para a redução de custos de projeto e para o ganho no tempo de desenvolvimento do sistema. Diversas tecnologias, como mineração de dados, aprendizado de máquinas, taxonomia e ontologias podem ser utilizadas para que se possa construir uma ferramenta automatizada de auxílio à análise de um software. Neste artigo é realizado um breve levantamento de técnicas que podem ser empregadas para o desenvolvimento de ferramentas automáticas que melhoram o processo de desenvolvimento de software e de algumas ferramentas já existentes, que são utilizadas para ajudar no desenvolvimento de software.
Palavras-chave: Dependabilidade de software, Análise de perigos, Análise de segurança.

Abstract
The use of tools to aid analysis of dependability and safety of software can help reduce project costs and the gain in system development time. Several technologies such as data mining, machine learning, taxonomy and ontologies can be used to build an automated tool to aid the analysis of software. In this paper a brief survey of techniques is conducted and they can be employed for the development of automated tools that improve the software development process, and some existing tools that are used to help in the development of software.
Keywords: Software dependability, Hazard analysis, Safety analysis.

Resumen
El uso de herramientas para ayudar al análisis de la fiabilidad y software de seguridad puede ayudar a reducir los costes del proyecto y la ganancia en el tiempo de desarrollo. Varias tecnologías como la minería de datos, aprendizaje automático, la taxonomía y la ontologías se pueden usar de modo que se puede construir una herramienta automatizada para facilitar el análisis de software. En este artículo se llevó a cabo una breve encuesta de técnicas que se pueden emplear para el desarrollo de herramientas automatizadas que mejoran el proceso de desarrollo de software, y algunas herramientas existentes que se utilizan para ayudar en el desarrollo de software.
Palabras clave: Fiabilidad de software, Análisis de peligros, Análisis de seguridad.

Introdução

Atualmente, a utilização de *software* em sistemas críticos tem se mostrado muito importante, pois sistemas do setor aeronáutico, energético e espacial apresentam muitas funcionalidades que possibilitam a utilização de programas computacionais para tratamento

dessas funções. Muitas destas funcionalidades, que eram desempenhadas por *hardware* de forma mecânica, agora são assimiladas por funções de *software*.

Apesar do *software* ter facilitado o desenvolvimento de aplicações críticas, o número de componentes envolvidos no sistema aumentou, o que pode fazer com que faltas e falhas ocorram em alguma parte do sistema. Muitas vezes, a utilização da automatização no processo pode acarretar na perda de informação ou na utilização de uma informação incorreta, o que pode ocasionar acidentes, causando danos materiais, físicos e prejuízos financeiros. Muitas vezes, os erros de projeto ou de desenvolvimento podem não ser visíveis durante o desenvolvimento, porém quando o *software* é colocado em operação, o erro pode se manifestar de modo catastrófico.

Diversas pesquisas mostram que problemas decorrentes de falhas em sistemas espaciais comprometem ou até mesmo inviabilizam a missão. Leveson (2004, 2009) descreve diversos acidentes que ocorreram com veículos espaciais relacionados a *software*. A autora levanta diversos fatores que causaram os acidentes, entre eles, ausência de requisitos, valores de variáveis incorretamente atribuídas, falta de validação de parâmetros de entrada, problemas de temporização e de testes, problemas de documentação, entre outros. Além disso, a autora apresenta um conjunto de lições aprendidas que podem ser utilizadas como referências para novas pesquisas.

De acordo com Lutz (2011), uma grande quantidade de *software* apoia a exploração espacial, com utilização no controle de órbitas, navegação de sondas exploratórias, *software* embarcado para captura de dados e outros. A engenharia de *software* possui um papel importante, oferecendo ferramentas, técnicas e abordagens sistemáticas para desenvolvimento de *software* espacial, visando garantir o funcionamento seguro e correto da missão espacial.

Em Véras et. al (2010), os autores apresentam um estudo que mostra erros encontrados nos requisitos de *software* espacial. Os resultados indicam uma alta taxa desses tipos de erros (9,5 %), e isso mostra que a análise de requisitos é importante e necessária para garantir o sucesso da missão.

Técnicas utilizadas em *hardware*, como FTA, FMECA e FMEA são adaptadas para *software* com o intuito de se melhorar o processo de desenvolvimento do *software*, analisando as principais falhas que podem ocorrer e os meios de se mitigar essas falhas.

A utilização de Inteligência Artifical (IA) no desenvolvimento de aplicações críticas para missões espaciais também é utilizada, permitindo assim que técnicas como descoberta do conhecimento, *data mining*, ontologias e *web* semântica sejam utilizadas para se descobrir informações a partir de dados já existentes e conhecidos. Shafto e Sierhuis (2010) apresentam um levantamento da utilização de IA em aplicações espaciais, fornecendo uma visão temporal (passado, presente e futuro) de como essas técnicas foram e podem ser utilizadas para melhorar as missões espaciais.

Este artigo tem como principal objetivo apresentar um breve levantamento de ferramentas que buscam automatizar o processo de análise de requisitos de um *software*, uma vez que vários problemas encontrados em projetos provem dos requisitos (LEVESON, 2004). Buscando atingir este objetivo, o artigo apresentada algumas ferramentas encontradas na literatura, como foram construídas para automatizar o processo e o modo como elas auxiliam na análise de requisitos. Ao final são apresentadas as considerações.

1 Materiais e Métodos

Para realizar o levantamento de ferramentas que auxiliam no processo de análise de requisitos foram realizadas pesquisas na literatura com a finalidade de se encontrar trabalhos que exploram a utilização de técnicas de IA para descoberta de novas informações e de aplicações automáticas que auxiliam na análise de falhas.

Os principais termos utilizados na pesquisa são decorrentes de técnicas empregadas na descoberta de conhecimento. Estas técnicas envolvem a utilização de dados já conhecidos para buscar informações desconhecidas que possam ser utilizadas para melhorar o processo do sistema.

Baseado nesse cenário, o artigo apresenta alguns trabalhos da área que utilizam IA para auxiliar no processo de melhoria de desenvolvimento de *software*, bem como ferramentas automáticas para melhoria dos requisitos dos sistemas. A seção 2 apresenta as ferramentas levantadas e as contribuições de cada uma delas para a análise de segurança de *software*.

2 Ferramentas Levantadas

Nesta seção são descritas as ferramentas levantadas durante a pesquisa na literatura. Um descritivo de cada uma delas e o modo como funcionam são apresentados para que se possa determinar as contribuições de cada uma para a análise de segurança.

2.1 *Predicting software faults in large space systems using machine learning techniques* (TWALA, 2011)

Nesse artigo, o autor apresenta algumas técnicas de aprendizado de máquina e em seguida aplica essas técnicas para prever falhas de *software* em sistemas espaciais. Uma das considerações feitas pelo autor é que a aplicação dessas técnicas requer uma grande quantidade de dados do *software* para que se possa treinar o modelo, pois as técnicas dependem de dados do passado para garantir melhor precisão na identificação de falhas. Muitas vezes esses dados não estão disponíveis ou não existem.

O autor aplicou cinco técnicas no trabalho:

Apriori → algoritmo utilizado para aprender regras de associação. Ele encontra regras que expressam relacionamentos probabilísticos entre itens em conjuntos de itens frequentes.

Árvores de decisão → Utilizadas para classificação e predição, têm como objetivo classificar corretamente o maior número de amostras do treinamento, bem como generalizar as

amostras de modo que dados invisíveis ou novos possam ser classificados com a maior precisão possível.

K vizinhos mais próximos → Busca proximidade de informações entre vizinhos próximos. Trabalha categorizando um item por meio da classificação do vizinho mais próximo já identificado.

Classificador Naive Bayes → Provê aprendizado combinando novos dados com dados já observados. Utiliza teoria da probabilidade para classificar conceitos.

Máquinas de vetores de suporte → Utilizada para reconhecimento de padrões, são classificadores de padrões que podem ser expressados na forma de hiperplanos para distinguir itens positivos de negativos.

A aplicação dessas técnicas foi realizada em um experimento composto por quatro conjuntos de dados, sendo três conjuntos coletados do repositório da NASA *metric data program* e um da base de dados de anomalias do JPL (*Jet Propulsion Laboratory*), da NASA. O modelo de predição que utiliza os classificadores foi construído utilizando a ferramenta WEKA[1].

Os resultados do experimento mostram que o melhor classificador foi o de Naive Bayes, conforme pode ser visto na Figura 1.

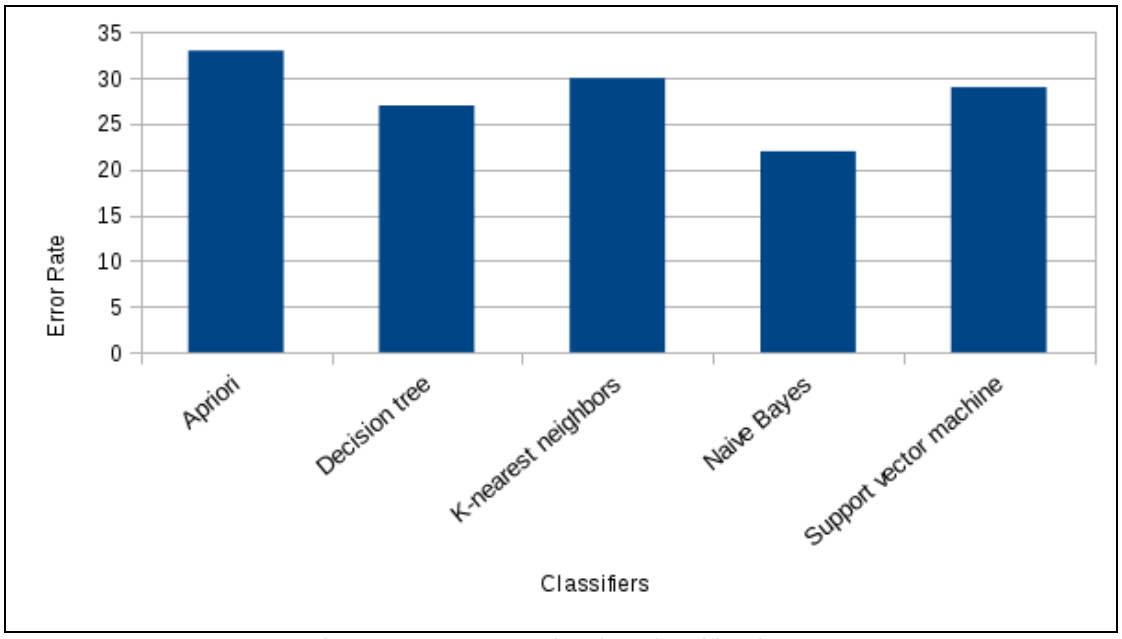

Figura 1: Desempenho dos classificadores
Fonte: Adaptado de Twala (2011)

Com os resultados obtidos, os autores afirmam que algoritmos de aprendizado de máquinas podem ser aplicados com sucesso na predição de falhas de *software* com múltiplos classificadores, sendo utilizados em conjunto. Individualmente, o classificador mais efetivo foi o Naive Bayes.

1 Ferramenta *open source* desenvolvida em Java para classificação de dados, regressão, regras de associação e visualização.

2.1 *A reliability, maintainability, and safety model to support the assessment of space vehicles* (RUIZ-TORRES et al., 2010)

Nesse trabalho, os autores apresentam um modelo para levantar os requisitos de confiabilidade e disponibilidade. Seu objetivo é fornecer estimativas de alto nível para segurança e manutenção de sistemas espaciais futuros.

Os autores apresentam um modelo RMS (*Reliability, Maintainability and Safety*) que utiliza os princípios básicos de confiabilidade para estimar as probabilidades de uma missão segura e a necessidade de reparos ou substituição durante o processo em solo, antes do lançamento e início da missão. Essa estimativa é baseada nas características do sistema principal do veículo, como número de subsistemas, tempo médio para reparo e a confiabilidade média de cada subsistema.

O modelo proposto foi desenvolvido no Microsoft Excel com funções escritas em Visual Basic para aplicações. A interface do *software* e suas funções foram desenvolvidas com a colaboração de engenheiros e gerentes da NASA. Essa interface pode ser visualizada na Figura 2.

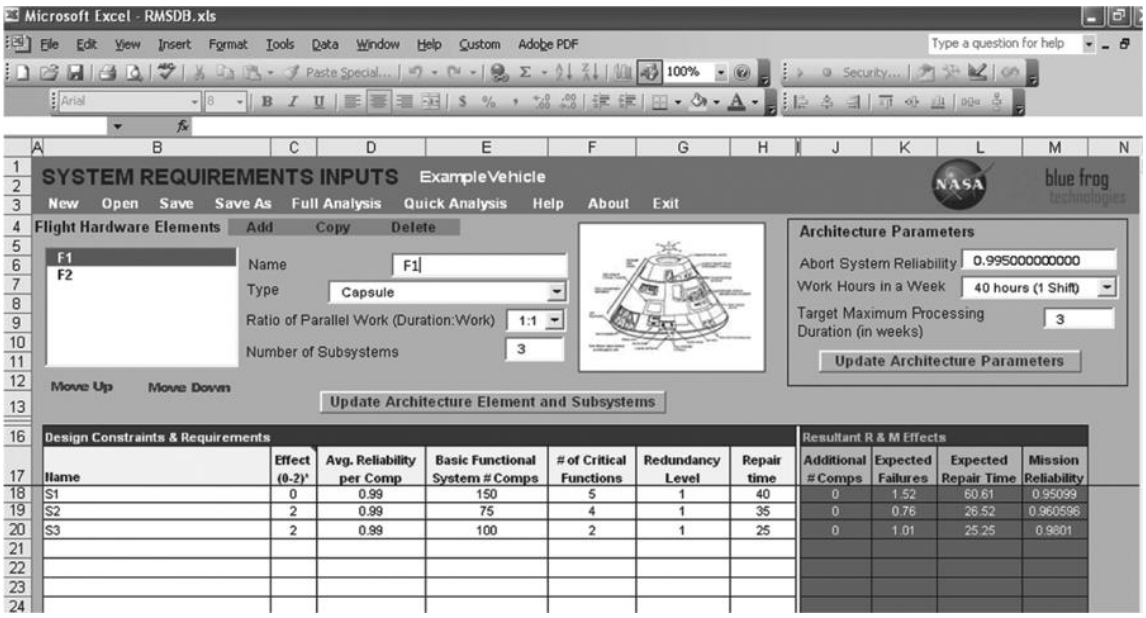

Figura 2: Interface do RMSDB
Fonte: Ruiz-Torres et al. (2010)

Com a ferramenta, os projetistas e engenheiros podem avaliar rapidamente seus projetos para segurança e manutenção. A ferramenta provê vários relatórios que auxiliam os projetistas a entender o efeito de decisões de confiabilidade e os subsistemas que devem receber maior atenção para melhoria da confiabilidade. As Figura 3 e 4 apresentam modelos de relatórios gerados pela ferramenta.

Revista Científica On-line
Tecnologia – Gestão – Humanismo
ISSN: 2238-5819

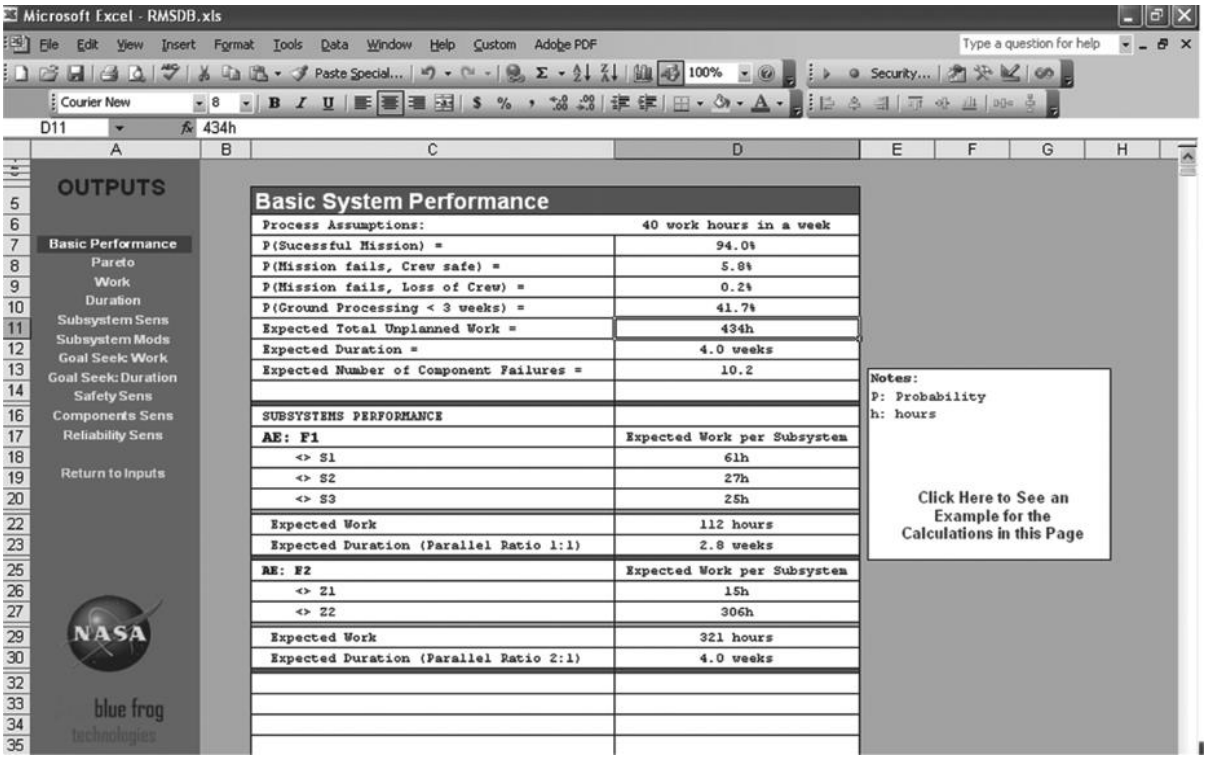

Figura 3: Relatório de desempenho de um sistema
Fonte: Ruiz-Torres et al. (2010)

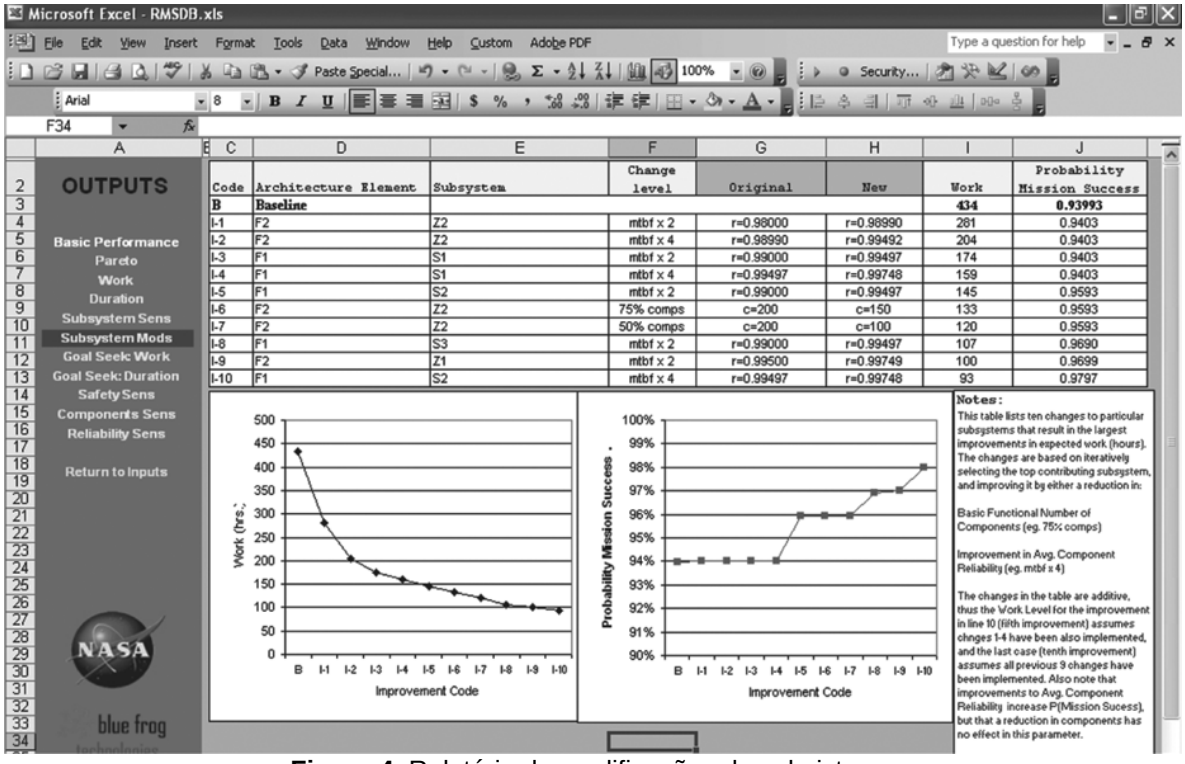

Figura 4: Relatório de modificações do subsistema
Fonte: Ruiz-Torres et al. (2010)

2.3 *NuFTA: A CASE tool for automatic software fault tree analysis* (YUN, LEE e YOO, 2010)

Nesse artigo, é apresentada uma ferramenta que gera automaticamente uma SFTA (*Software Fault Tree Analysis*). Essa ferramenta CASE utiliza como entrada uma especificação

formal de requisitos NuSCR. NuSCR é uma linguagem de especificação formal utilizada para detalhar requisitos de *software* do sistema de proteção do reator nuclear da Korea.

Os autores explicam que a ferramenta gera as SFTAs a partir de três *templates* pré-definidos. Estes *templates* não são discutidos no artigo, porém são apresentados no trabalho de Kim (2005).

A ferramenta foi desenvolvida em Java e gera a SFTA a partir da especificação NuSCR em XML. Em seguida, a SFTA é gerada e armazenada em um arquivo XML. Também é criada uma fórmula lógica que representa a SFTA. A Figura 5 apresenta uma tela do *software* com uma SFTA já criada a partir da especificação formal.

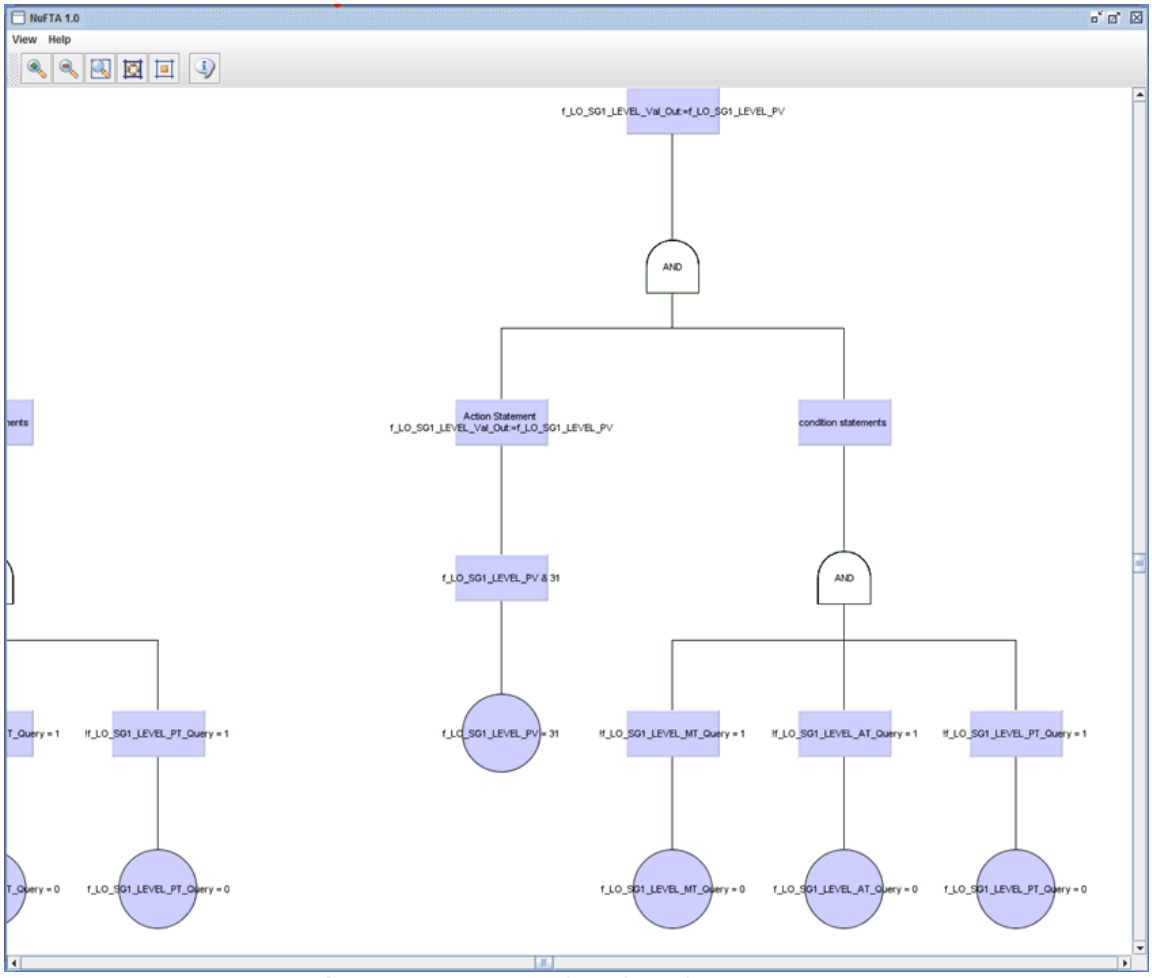

Figura 5: SFTA gerada pelo *software* NuFTA
Fonte: Yoon, Jo e Yoo (2011)

2.4 *Conducting FMEA over the software development process* (GEORGIEVA, 2010)

Nesse trabalho, a autora apresenta uma ferramenta *web* para geração de uma SFMEA. Ela afirma que a FMEA é amplamente utilizada como método de análise para estabelecer confiança em sistemas, porém ela não é muito utilizada na engenharia de *software*, por isso é proposta a aplicação da FMEA para o processo de desenvolvimento de *software*.

No artigo é descrito todo o processo de construção de uma FMEA, sendo dividido em oito passos. Para aplicação da FMEA em um *software*, a autora adota a norma ISO 12207 para definir o processo de desenvolvimento de *software*. Baseado na norma e em experiências próprias, foram definidos oito estágios do processo de desenvolvimento de *software*: 1) Planejamento e gerenciamento do projeto; 2) Aquisição dos requisitos do *software*; 3) Projeto do sistema – arquitetural e projeto detalhado; 4) Implementação e codificação do *software*; 5) Teste do programa e integração do sistema; 6) Teste de qualificação e entrega do *software*; 7) Instalação do *software* e manutenção; 8) Aceite do *software* e suporte.

Para cada um dos estágios, são levantados os possíveis modos de falhas. A ferramenta desenvolvida (Figura 6) auxilia na análise FMEA de um projeto já existente ou de um novo projeto.

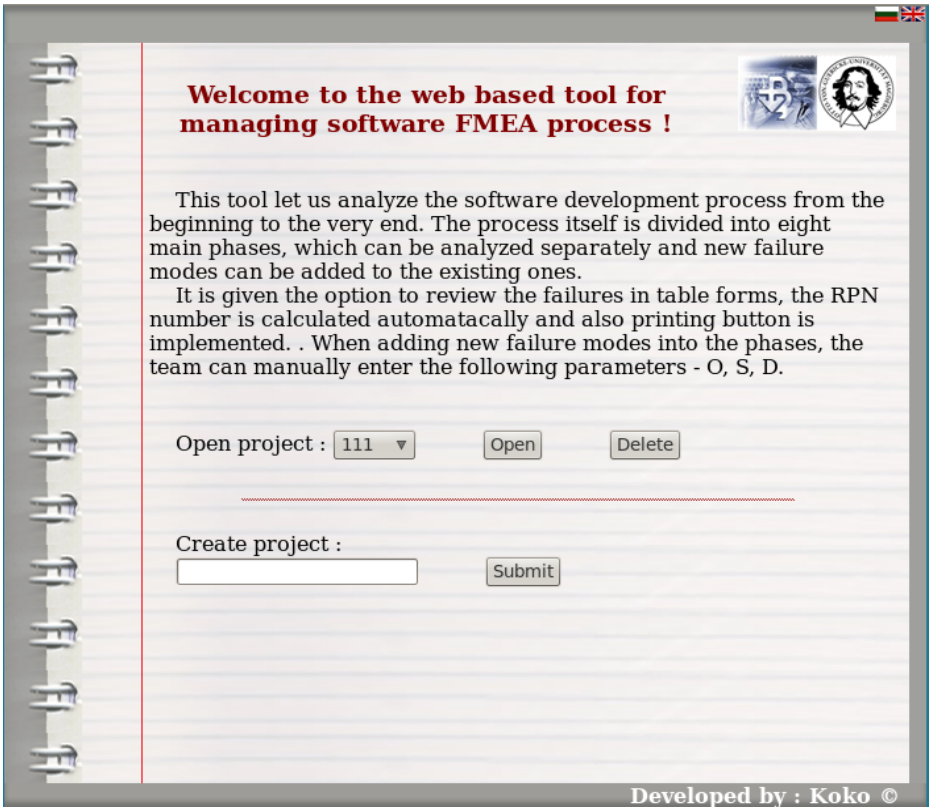

Figura 6: Ferramenta para análise FMEA
Fonte: Georgieva (2010)

A autora indica que é possível alterar os modos de falhas, adicionar novas falhas e novas recomendações e analisar o processo completo de desenvolvimento. A Figura 7 apresenta uma tela em que é possível alterar as informações de uma parte do projeto. Uma das grandes vantagens da utilização da ferramenta é a possibilidade de acompanhar todas as fases do projeto, todas as falhas prováveis e as recomendações para elas.

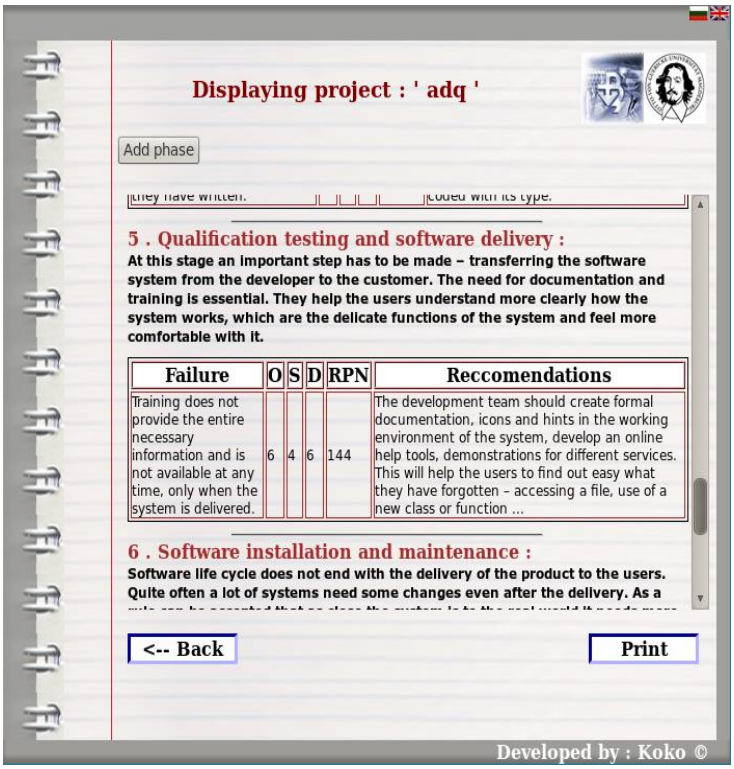

Figura 7: Ferramenta da FMEA com informações que podem ser alteradas
Fonte: Georgieva (2010)

2.5 *XML technology planning database: lessons learned* (SOME; NEFF , 2005)

Nesse artigo, os autores apresentam um banco de dados hierárquico baseado em XML denominado XCALIBR (*XML Capability Analysis LIBRary*). O banco de dados contém os requisitos da missão e as capacidades tecnológicas, que são relacionadas para uso de um dicionário XML. Esse dicionário codifica uma taxonomia para missões espaciais, sistemas, subsistemas e tecnologias. Também são apresentadas no artigo, as lições aprendidas durante a construção e teste do protótipo do banco de dados e a motivação para migrar a taxonomia em XML para uma ontologia.

O XCALIBR utiliza uma estrutura em árvore, que descreve toda organização da NASA e funções e estruturas das missões espaciais. A taxonomia, conforme visto na Figura 8, oferece uma decomposição hierárquica de cada seção do banco de dados.

A ferramenta possui uma interface gráfica para navegação, buscas e entrada de dados. Trabalha com um conjunto de ferramentas baseadas em Excel, utiliza a máquina de buscas Qexo (*processador Xquery open source*) para processar buscas no banco de dados. Possui a taxonomia detalhada da organização (Figuras 8 e 9), que contém as informações para que o banco de dados defina dados qualitativos, quantitativos e os relacionamentos. Por fim, a ferramenta utiliza o Tamino XML *Server* para armazenar, gerenciar, publicar e trocar documentos XML em seu formato nativo.

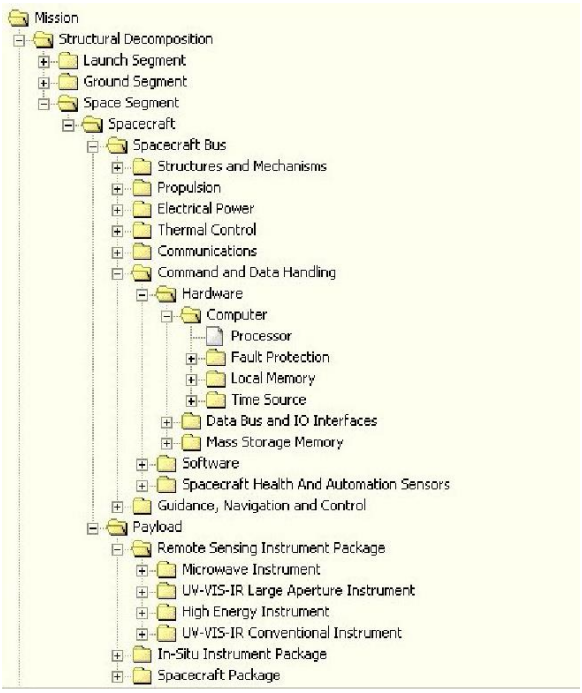

Figura 8: Parte da taxonomia do XCALIBR
Fonte: Some e Neff (2005)

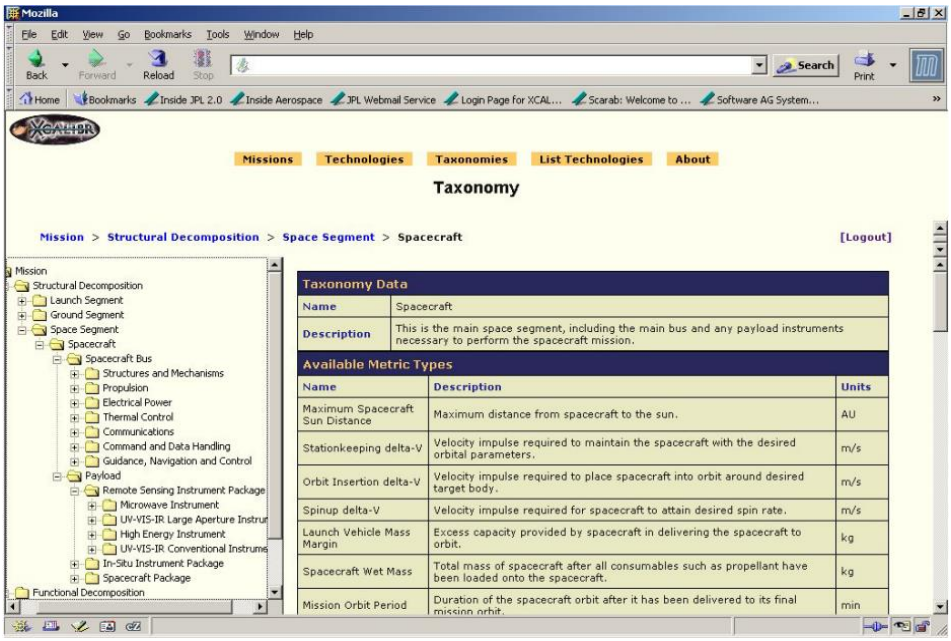

Figura 9: Taxonomia visualizada na interface gráfica
Fonte: Some e Neff (2005)

De uma maneira resumida, o XCALIBR é uma ferramenta que permite aos usuários especificar requisitos de missão e as capacidades tecnológicas desde o nível de sistema até o de componente. A Figura 10 apresenta a tela de requisitos de missão da ferramenta.

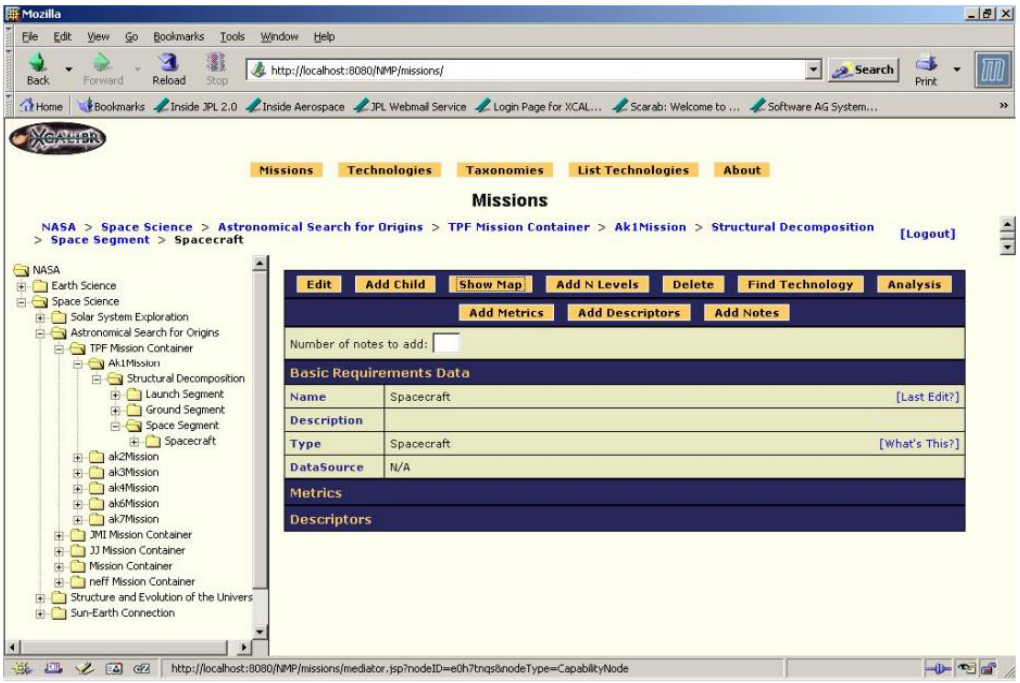

Figura 10: Interface para requisitos de missão
Fonte: Some e Neff (2005)

Como a ferramenta é baseada em Excel, é possível exportar os dados diretamente para planilhas, permitindo assim que se utilize este *software* para manipular os dados do projeto. Uma tela com dados exportados para o Excel é apresentada na Figura 11.

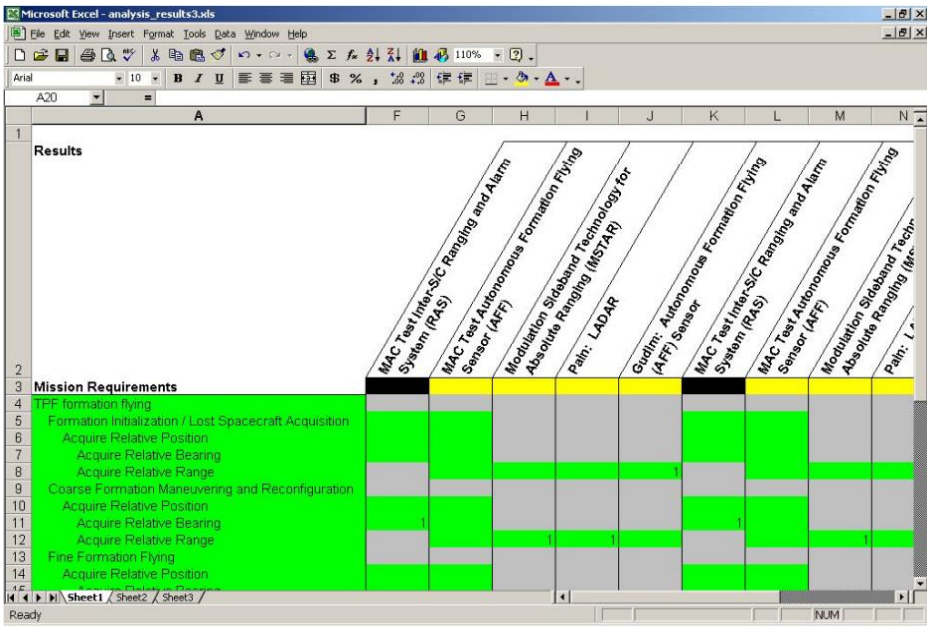

Figura 11: Análise dos dados no Excel
Fonte: Some e Neff (2005)

A partir de testes realizados para validar a usabilidade da ferramenta, os autores apresentaram algumas lições que podem auxiliar no desenvolvimento de novas ferramentas.

Com relação à navegação, os autores dizem que o sistema deve ser suficientemente flexível para suportar vários tipos de *views*. A interface deve ser óbvia e intuitiva, pois uma interface óbvia para um desenvolvedor pode não ser simples para um usuário. Segundo os autores, interfaces gráficas não são suficientes para ler uma grande quantidade de dados, sendo necessária a utilização de ferramentas específicas para esta atividade em conjunto com as facilidades da interface gráfica.

Com relação à taxonomia, os autores indicam que a migração para uma ontologia oferece muitas vantagens para os desenvolvedores e usuários do XCALIBR. Utilizar uma ontologia torna o sistema mais flexível, pois o esquema não precisa ser alterado quando novos relacionamentos e atributos forem adicionados. Uma taxonomia auxilia o usuário a entender e organizar as informações, enquanto uma ontologia pode ser construída para a leitura do indivíduo e ser compreensível por máquinas. Um protótipo da ontologia é apresentado na Figura 12.

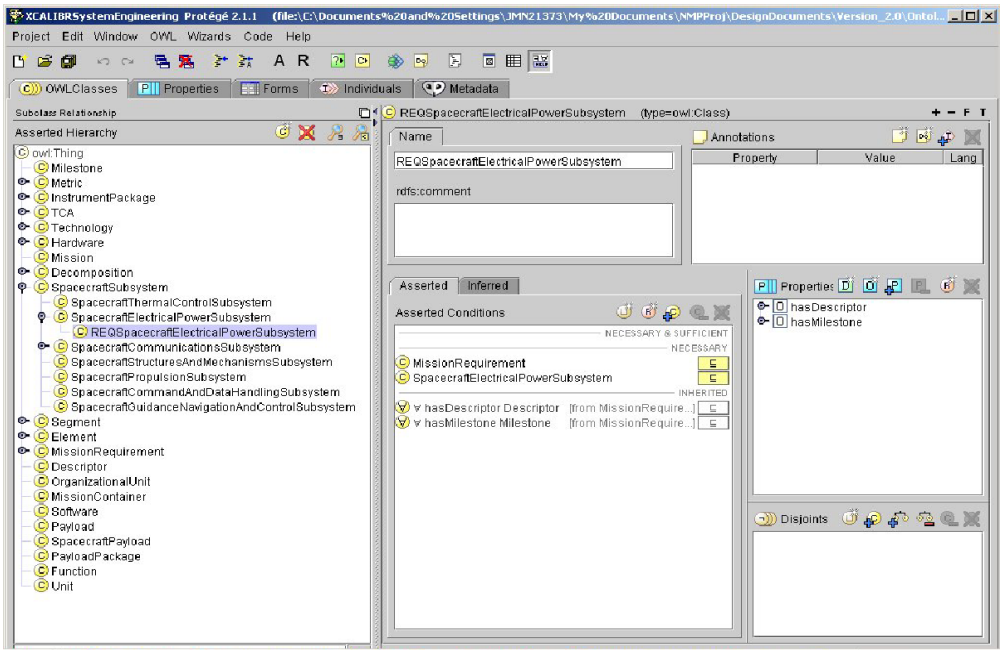

Figura 12: Protótipo da ontologia do XCALIBR
Fonte: Some e Neff (2005)

Considerações Finais

Técnicas conhecidas para análise de dependabilidade e segurança de *software*, como SFTA e SFMEA podem ser automatizadas para se ganhar tempo no desenvolvimento de *software*. Essas técnicas podem identificar, já na fase inicial do projeto, correções que precisam ser feitas, evitando que as falhas encontradas sejam propagadas durante a realização do projeto.

A utilização de técnicas de inteligência artificial, como mineração de dados, aprendizado de máquinas e ontologias podem auxiliar na análise de segurança, uma vez que é possível realizar buscas e descobrir novas informações a partir de um conjunto de dados já existentes. Dados de projetos anteriores, semelhantes ao que será desenvolvido, podem ser utilizados para

que a ferramenta de apoio descubra novas informações e assim auxilie na correta construção do projeto desde o início.

O desenvolvimento de novas ferramentas automatizadas para análise de dependabilidade e segurança de *software* pode auxiliar na melhoria do processo de desenvolvimento e ajudar na tomada de decisões do projeto. É importante ressaltar que a ferramenta não vai construir o projeto ou dizer como ele deve ser feito, apenas indicará possíveis soluções que o analista ou engenheiro poderá utilizar para a construção do sistema.

Referências

GEORGIEVA, K. Conducting fmea over the software development process. **ACM SIGSOFT Software Engineering Notes**, ACM, v. 35, n. 3, p. 1–5, 2010.

KIM, T. **Property-based theorem proving and template-based fault tree analysis of NuSCR requirements specification**. Tese (Doutorado) - Korea Advanced Institute of Science and Technology, 2005.

LEVESON, N. G. Role of software in spacecraft accidents. **Journal of spacecraft and Rockets**, v. 41, n. 4, p. 564–575, 2004.

LEVESON, N. G. Software challenges in achieving space safety. **British Interplanetary Society 62**, 2009.

LUTZ, R. Software engineering for space exploration. **Computer**, IEEE, v. 44, n. 10, p. 41–46, 2011.

RUIZ-TORRES, A. J.; ZHANG, J.; ZAPATA, E.; PENNATHUR, A.; RHODES, R.; MCCLESKEY, C.; COWEN, M. A reliability, maintainability, and safety model to support the assessment of space vehicles. **International Journal of Quality & Reliability Management**, Emerald Group Publishing Limited, v. 27, n. 4, p. 486–504, 2010.

SHAFTO, M. G.; SIERHUIS, M. Ai space odyssey. **Intelligent Systems, IEEE**, IEEE, v. 25, n. 5, p. 16–19, 2010.

SOME, R.; NEFF, J. Xml technology planning database: lessons learned. In: IEEE. **IEEE Aerospace Conference**, p. 743–757, 2005.

TWALA, B. Predicting software faults in large space systems using machine learning techniques. **Defence Science Journal**, v. 61, n. 4, p. 306–316, 2011.

VÉRAS, P. C.; VILLANI, E.; AMBROSIO, A. M.; SILVA, N.; VIEIRA, M.; MADEIRA, H. Errors on space software requirements: a field study and application scenarios. In: IEEE. **IEEE 21st International Symposium on Software Reliability Engineering (ISSRE)**, p. 61–70, 2010.

YOON, S.; JO, J.; YOO, J. A domain-specific safety analysis for digital nuclear plant protection systems. In: IEEE. **Fifth International Conference on Secure Software Integration & Reliability Improvement Companion (SSIRI-C)**, p. 68-75, 2011.

YUN, S.; LEE, D.-A.; YOO, J. Nufta: a case tool for automatic software fault tree analysis. In: **Transactions of the Korean Nuclear Society Spring Meeting**, 2010.

Recebido em 08/04/2016

Aceito em 31/10/2016