UMA DISCUSSÃO SOB O PROCESSO DE VERIFICAÇÃO E VALIDAÇÃO DE SOFTWARE CRÍTICO E SEGURO; DEFINIÇÃO DE CONCEITOS, LIMITAÇÕES, NORMAS, OBJETIVOS E TÉCNICAS APLICADAS.

LORENA GAYARRE PEÑA[1], MARCELO LOPES DE OLIVEIRA E SOUZA[2].

1. *Sala 20 prédio satélite, Departamento ETE, INPE*
Avda. Astronautas 1758 12227-010 SJC SP Brasil
E-mails: lorenagayarre@gmail.com

2. *Sala 49 prédio satélite, Departamento ETE, INPE*
Avda. Astronautas 1758 12227-010 SJC SP Brasil
E-mails: marcelo@dem.inpe.br

Abstract— Software verification and validation (V&V) became extremely important on the software development life cycle when safety-critical software appeared. This software is characterized because its failure causes huge losses, even it can cause human life losses. In this paper, an overview about safety-critical software validation and verification is done; items such as definitions, misconceptions, limitations, standards, hazard levels and techniques among others, are addressed.

Keywords— Verification and validation, software life cycle, safety-critical software, techniques, standards.

Resumo— O processo de verificação e validação de software voltou a ser uma fase extremamente importante no ciclo de vida de desenvolvimento de software com a aparição do software crítico e seguro. Este software é caracterizado porque uma falha nele pode causar grandes perdas, incluso perda de vidas humanas. Neste artigo, uma discussão sobre o processo de verificação e validação de software crítico e seguro é feita, são tratadas as definições, erros de conceito, limitações, normas, níveis de condição de falha e técnicas, entre outros.

Palavras-chave— Verificação e validação, Ciclo de vida do software, software crítico e seguro, técnicas, normas e padrões.

1 Introdução

O processo de verificação e validação (V&V) e, por extensão, certificação, vem se tornando uma fase extremamente importante do ciclo de vida de um projeto, dada a aparição do software crítico e seguro (*safety-critical software*), *i.e.* software cuja falha pode causar perdas humanas (vide seção 6).

Segundo o glossário standard de IEEE aplicado a engenharia de software, o processo de verificação e validação é uma fase que determina se:

- Os requisitos para um sistema ou software estão completos e corretos;
- Os resultados de cada fase de desenvolvimento cumprem os requisitos ou condições impostos pela fase prévia;
- O sistema ou software final cumpre com as especificações.

Igualmente, o processo de certificação determina mediante uma autoridade se o produto respeita as normas correspondentes segundo a aplicação do mesmo (vide seção 6).

Dentro do processo de V&V podem se diferenciar duas fases ou ações; a verificação e a validação. A verificação é a ação de comprovar se o produto cumpre com as especificações de projeto. Validação é a ação de constatar se o produto realmente se comporta como esperado, i.e. responde às exigências do usuário final. Ambos os processos são igualmente importantes, já que um produto final verificado (i.e. ele atinge as especificações de projeto), mas não validado (i.e. não responde aos requisitos de usuário), não seria útil; do mesmo modo, se o produto responde aos requisitos de usuário (validado), mas não satisfaz as especificações (não verificado), ele também não pode ser utilizado, pois não será certificado pelas autoridades. A tabela 1 apresenta as diferenças entre ambas.

Tabela 1. Diferenças entre verificação e validação.

VERIFICAÇÃO	VALIDAÇÃO
É um processo estático de verificação de documentos, projeto e código.	È um processo dinâmico que valida / testa o produto final.
Não requer execução de código.	Requer execução de código.
É um trabalho manual, baseado na inspeção de documentos.	É um trabalho automatizado baseado na execução do programa.
O objetivo deste processo é comprovar que o software segue as especificações.	O objetivo deste processo é comprovar que o software se comporta como esperado.
Usa métodos ou técnicas como inspeções, walkthroughs, revisão de código, entre outras. (Vide seção 7)	Usa técnicas como testes de caixa preta e testes de caixa branca a diferentes níveis; unidade, integração, sistema, entre outros. (Vide seção 7)
Geralmente, este processo se produz antes da validação.	Este processo se produz depois da verificação.
Responde à questão: Está-se desenvolvendo o corretamente?	Responde à questão: Está-se desenvolvendo o produto correto?

Acha erros relacionados com a definição de projeto.	Acha erros relacionados com o funcionamento do produto.

2 Limitações do processo de V&V

O processo de verificação e validação tem como objetivo comprovar que o produto está livre de erros e atinge as expectativas do usuário final. Este objetivo é só teórico, já que resulta inalcançável na prática. Nesta seção as limitações que fazem com que a meta deste processo seja uma utopia, segundo Collofelo (1988), são descritas:

1. Fundamentos teóricos: Não existe nenhum processo de teste ou análise que possa ser usado para garantir exatidão do produto, i.e., nenhum produto final está isento de erros. Mas ao se aplicar V&V, tenta-se assegurar que a máxima quantidade de erros possíveis foram encontrados e corrigidos.
2. Inviabilidade de testar todos os dados: Na maioria dos programas é impossível testar todas as entradas; devido à grande quantidade de combinações possíveis seria inviável tanto temporal quanto economicamente.
3. Inviabilidade de testar todos os caminhos do software: Na maioria dos códigos, é impossível testar todas as opções de execução devido à grande quantidade de combinações possíveis.
4. Não existe prova de exatidão absoluta: Não existe uma prova aplicável para comprovar que as expectativas do usuário foram devidamente atingidas; para isso deveria existir um algoritmo formal que representasse exatamente esses requisitos de usuário, que é inalcançável também.

3 Conceitos errados em V&V

Habitualmente alguns conceitos relacionados com verificação e validação são misturados, errados ou simplesmente desconhecidos.

Nesta seção, os conceitos básicos do processo de verificação e validação são esclarecidos e uma comparação entre conceitos tipicamente misturados é feita.

- *Ciclo de vida de desenvolvimento de um produto*: Define o conjunto de fases seguidas desde que o produto é concebido até que ele é retirado do mercado. Existem vários modelos, ver L. Gayarre (2013), mas todos eles estão baseados em quatro fases gerais; (1) definição de requisitos de usuário e especificações de projeto; (2) designação do projeto mediante diagramas e documentação; (3) desenvolvimento do produto; e (4) processo de verificação e validação. Um exemplo de modelo, e o mais aplicado no desenvolvimento de software crítico e seguro, é o modelo em V, mostrado na figura 1.

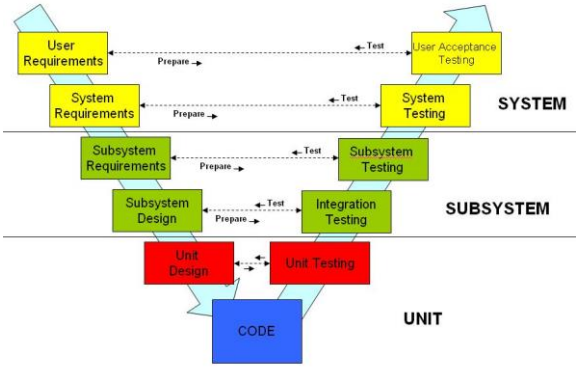

Figura 1. Modelo em V.

- *Verificação*: É o processo levado a cabo para assegurar que o produto é internamente consistente (N. Jenkins, 2008), i.e. que o produto satisfaz os testes, que os testes seguem as especificações, que as especificações atingem os requisitos, e assim por diante.
- *Validação*: Usa fontes externas como referência para confirmar que o produto atinge as expectativas de usuário, i.e. que ele se comporta como deveria.
- *V&V*: O objetivo do processo de verificação e validação é eliminar a máxima quantidade de erros acumulados durante o ciclo de vida do produto, assegurando que as especificações são cumpridas e os objetivos atingidos. Este processo começa junto com a especificação de requisitos no ciclo de vida do software e acompanha durante todo o desenvolvimento.
- *Certificação*: É um processo levado a cabo normalmente por agências do governo ou organizações reconhecidas, no qual o produto é revisado para garantir que o processo de V&V seguiu as normas requeridas. Este processo pode ser aplicado em duas ocasiões: para fazer uma diferenciação da qualidade de um produto, ou para legitimar que, efetivamente, o produto seguiu um processo de desenvolvimento específico. No primeiro caso, a certificação só é usada para outorgar distinção ao produto. O segundo caso é aplicado quando é necessário por lei, como é o caso do software crítico e seguro.
- *VV&A*: O processo de verificação, validação e certificação (VV&A em inglês) é aplicado normalmente a sistemas ou software que requerem cumprir umas normas para serem comercializados, por exemplo, em software crítico e seguro (vide seção 5). Este processo cobre todo o ciclo de desenvolvimento do produto, assegurando que se cumprem os objetivos de V&V. (vide seção 4).
- *Q&A*: (Quality assurance) Este processo é, as vezes, confundido com o processo de V&V. Ambos os dois processos se complementam,

sendo Q&A o que determina se o projeto segue os procedimentos especificados e não sai da linha de desenvolvimento definida.

- *Testes*: O processo de teste é uma parte do processo geral de verificação e validação. Consiste em introduzir entradas no sistema e estudar suas saídas, comparando os dados reais com os teoricamente esperados. Normalmente o processo de teste fica dentro do processo de validação.
- *Testes funcionais e não funcionais*: Os testes funcionais estudam o comportamento do software para se atingir às especificações; frente a um banco de dados de entrada revisa se as saídas são as esperadas teoricamente. Os testes não funcionais são aqueles que se focalizam nas características de desempenho da aplicação de software, como tempo de demora ou complexidade do código, entre outras.
- *Requisitos de usuário VS especificações de projeto*: Os requisitos de usuário são as expectativas de usuário; aquilo que o usuário gostaria que o produto fosse. As especificações de projeto são o conjunto de restrições ou características que o produto realmente vai cumprir. Destes dois conceitos, na primeira fase do ciclo de vida do produto, nascem dois documentos; de requisitos de usuário e de especificações de projeto. Se bem que as vezes estes documentos são os mesmos, pois o objetivo do produto é atingir as expectativas de usuário, isto não é sempre assim. Por exemplo, na indústria telefônica, muitas vezes o celulares possuem mais capacidades do que o usuário realmente precisa. De outro lado, o atendimento ao cliente não é sempre tão bom quanto o usuário queria.
- *Software crítico e seguro*: Se considera software crítico e seguro aquele cuja falha pode causar mortes humanas (ver seção 6). Este software deve cumprir com as normas relativas à área de aplicação, como DO-178B/C para aeronáutica ou ISO 26262 para a indústria automotiva (vide seção 5).

4 Objetivos de V&V

A função de um processo de V&V é satisfazer os requisitos de usuário, aliás, para medir se isso foi alcançado, certos objetivos devem ser definidos. Segundo Collofelo (1988), podem se nomear muitos objetivos do processo de V&V tais como segurança, portabilidade, usabilidade, manutenção, entre outras. Mas todos os objetivos não podem ser atingidos sempre; estes vão depender de cada produto e devem ser definidos no início do projeto. Nesta seção só os objetivos principais do processo de verificação e validação desde o ponto de vista de funcionalidade e desempenho são apresentados; eles são cinco:

1. Exatidão: O produto está isento de erros.
2. Consistência: O produto é consistente com ele mesmo e com outros produtos.
3. Necessidade: Todas as funcionalidades do produto são necessárias para atingir as especificações, i.e, não existe funcionalidade que não tenha requisito associado.
4. Suficiência: O produto está completo, i.e. não existe requisito que não tenha funcionalidade associada.
5. Desempenho: O produto satisfaz os requisitos não funcionais, i.e. requisitos de desempenho.

5 Normas e padrões para software crítico e seguro

Existem várias áreas de aplicação onde, nas últimas décadas, os sistemas de software tem-se desenvolvido muito e tomado vantagem frente outros tipos de sistemas.

Algumas destas áreas precisam de sistemas de qualidade certificada, pois o funcionamento deles é crítico para o sucesso da missão, chamando de missão o objetivo para o que o software foi implementado. Para esta qualidade ser reconhecida, existem normas ou padrões criados por organizações nacional/internacionalmente reconhecidas ou mesmo por governos. Estas normas descrevem como realizar o processo de desenvolvimento do produto para poder assegurar que certo nível de qualidade está sendo atingido.

Seguir estes padrões é obrigatório em algumas áreas de atuação como a indústria automobilística, a indústria aeronáutica, a indústria ferroviária, entre outras.

A tabela 2 recolhe um resumo de algumas indústrias que usam software crítico e seguro e as normas reconhecidas/usadas internacionalmente (vide Mainar, 2009).

Tabela 2. Classificação de indústrias e normas aplicáveis.

INDÚSTRIA	NORMA / DESCRIÇÃO
AERONÁUTICA	DO-178B e revisão DO-178C
AUTOMOBILÍSTICA	ISO 26262
MÉDICA	IEC 62304
ENERGIA NUCLEAR	IEC 60880
FERROVIÁRIA	EN 50128

6 Níveis de software em V&V

Os produtos de software seguros e críticos podem ser classificados dependendo da chamada condição de falha, isto é, em função da severidade do efeito da falha. Segundo a classificação de severidade recomendada pela norma MIL-STD-1629A:

Tabela 3. Classificação de SW segundo a condição de falha.

Condição de Falha	Descrição
Catastrófico (Catastrophic)	Uma falha que causa mortes ou perda do veículo ou arma.
Crítica (Critical)	Uma falha que pode causar grandes danos humanos severos, ou danificação do veículo ou sistema que causa perda da missão.
Marginal (Marginal)	Uma falha que causa danos menores e pode provocar atraso ou degradação da missão.
Menor (Minor)	Uma falha que não provoca danos, mas deve ser solucionada.

Em função da classificação do software e da área de aplicação, uma norma diferente deverá ser utilizada e, para poder demonstrar frente às autoridades certificadoras que essas normas foram seguidas, um conjunto de documentos específicos devem ser gerados e apresentados às próprias autoridades.

Como exemplo, na área de aeronáutica, a norma DO-178B e a conseqüente revisão DO-178C são aplicadas. Estas normas classificam o software em cinco níveis de severidade (Catastrófica, perigosa, maior, menor e sem efeito); segundo o nível onde esse software embarcado está, diferentes documentos devem ser gerados e apresentados à organização para levar a cabo o processo de certificação.

7 Técnicas de V&V

Existem muitas técnicas para verificar e validar um software. Elas são escolhidas em função do tipo (e.g. modelo de simulação, software embarcado) ou da indústria de aplicação (e.g. aeronáutica, automobilística) entre outros.

Estas técnicas também serão escolhidas em função da análise de risco do software (ver seção 6) e da norma aplicável (ver seção 5)

Nesta seção uma classificação das técnicas de V&V de software mais aplicadas será feita e a utilidade e aplicabilidade delas será explicada.

7.1 Classificação

A análise de software no processo de V&V pode ser classificada em dois grandes grupos, análise estática e análise dinâmica.

Análise estática

Este tipo de análise não precisa da execução do código, mas pode ser requerido executá-lo mentalmente. Ela avalia a exatidão e precisão do código fonte. Para esta análise existem técnicas tanto manuais como automatizadas.

Entre as técnicas manuais, conhecidas também como revisões, cabe destacar as seguintes (ver A. Stellman (2005) e ESA (1995)):

- Deskcheck: É a técnica menos formal. Nela, o desenvolvedor envia um produto ou documento aos colegas de equipe de trabalho para eles revisarem e anotarem os erros foram encontrados. O processo não é registrado, é só um trabalho interno.
- *Revisões técnicas*: Avaliam o progresso do software com respeito ao plano. Concentram os esforços mais na qualidade do processo do que nos detalhes da implementação. Estas revisões são realizadas ao final de cada fase para estudar os resultados e os produtos obtidos e decidir o começo da fase seguinte. Como saída deste processo, é gerado um relatório que descreve o estado do projeto, as ações seguintes e as pessoas que participaram da revisão, entre outras.
- *Walkthroughs*: O objetivo é detectar e documentar erros, violações das normas ou qualquer outro problema. Não é um trabalho do desenvolvedor de código, mas de toda a equipe de projeto para achar quanto mais erros ou desvio dos padrões, melhor. É um processo pouco formal; não precisa de uma preparação para levar a cabo o processo e o falado na reunião não é registrado, só se registram os erros para poder solucioná-los.
- *Inspeções*: Igual às duas técnicas anteriores, uma inspeção tem como objetivo examinar o produto numa determinada fase do projeto e identificar erros. Este processo é mais formal; consiste em cinco fases, primeiro todos os participantes recebem a documentação a ser inspecionada. Individualmente essa informação é lida e preparada. No dia da reunião os problemas são identificados e registrados num relatório que é entregue ao time de desenvolvimento. Por último, o time soluciona os problemas achados e documenta as soluções tomadas. Uma inspeção é um processo mais formal do que um walkthrough e demora mais, mas é também muito mais efetivo para detectar falhas e resolver elas nas fases iniciais do projeto economizando tempo e dinheiro.
- *Auditorias*: É uma técnica de verificação que estuda a adequação dos processos às normas e padrões. Este processo deve ser realizado por uma equipe alheia ao desenvolvimento do produto. É uma técnica formal para comprovar que o processo de desenvolvimento está seguindo as normas, padrões correspondentes, que os documentos necessários estão sendo gerados, que os testes estão sendo realizados e que os resultados estão sendo registrados. Habitualmente, se alguma norma está sendo seguida, é a agenda da norma que descreve em que momento as auditorias devem ser feitas, quais os resultados ou documentos que devem ser entregues e quais os objetivos da auditoria.

As técnicas automatizadas estão baseadas no cumprimento de normas para escritura de código fonte ("*coding standard*"). Estas normas asseguram uma maior qualidade do código fonte e eliminam erros típicos devido à má implementação. Algumas

normas são MISRA-C, MISRA-C++ ou JSF++. Existem ferramentas de análise estática de código, tais como Klocwork, que inclui um corretor automático de código no ambiente de implementação, ou LDRA que permite criar a própria norma em função dos interesses de projeto (i.e. se uma das regras da norma MISRA-C é muito severa para o código que está sendo escrito, e o projeto não precisa cumprir essa norma, tem a possibilidade de criar uma norma específica para o projeto em particular). A figura 2 mostra uma captura de tela da ferramenta LDRA no momento de escolha da norma aplicada na análise estática.

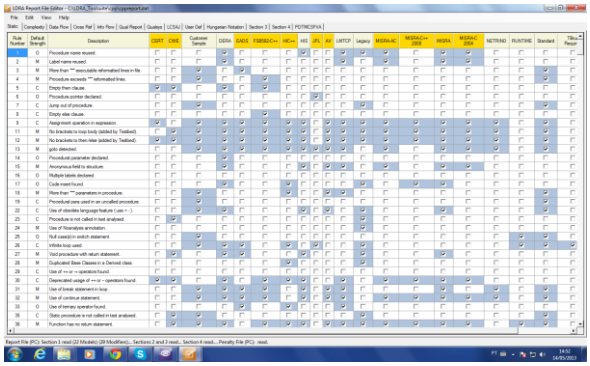

Figura 2. Normas para análise estática com a ferramenta LDRA.
Figura doada pela empresa Konatus (2013)

Análise dinâmica

Este tipo de análise é aplicada ao código em execução e o objetivo é estudar o comportamento do software em função das entradas aplicadas e comparar as saídas obtidas com as teoricamente esperadas. As técnicas aplicadas na análise dinâmica são conhecidas como testes e podem ser classificadas em dois tipos; os testes de caixa preta e os testes de caixa branca.

Testes de caixa preta

Nos testes de caixa preta o código é tratado como um pacote monolítico que recebe as entradas e devolve as saídas. Neste tipo de testes não importa como o software atua, mas só a precisão e exatidão das saídas. Este tipo de testes é conhecido também como testes funcionais. As técnicas mais conhecidas de caixa preta são:

- Classes de equivalência: Os dados de entrada são agrupados em classes de dados com um comportamento similar no software. Os resultados são usados como valores representativos de cada classe de possíveis dados de entrada.
- Valores limite: Este tipo de teste introduz os valores das bordas da classe (valores máximos e mínimos); o software é mais propenso a falha nestes valores por serem os limites implementados no código (e.g. nas funções de decisão como *if* ou *switch*, entre outras).
- Valores inválidos: Se injetam no sistema dados de entrada inválidos para comprovar se reconhece esses dados inválidos e responde a eles como deveria.

Testes de caixa branca

A finalidade dos testes de caixa branca é estudar a estrutura e as capacidades não funcionais do código. O objetivo é detectar erros sobre como o código foi implementado. A ideia principal e usar valores de entrada tais que o código completo seja executado; a seleção destes valores de entrada é crucial para alcançar bons resultados. Segundo RPG (2000), existem seis técnicas de caixa branca que são:

- *Branch test:* Detecta se todos os ramos do código foram executados.
- *Condition test:* Detecta se em cada rama de condição de código todas as condições foram executadas.
- *Data flow test:* Está baseado num tipo de diagrama chamado diagrama de controle de fluxo desenhado na fase de desenvolvimento de software. Detecta se todas as seqüências de eventos relacionadas com o estado das estruturas dos dados foram executadas.
- *Loop test:* Detecta se todos os loops foram executados.
- *Path test:* Está baseado num tipo de diagrama chamado diagrama de caminhos de execução, desenhado na fase de desenvolvimento de software. Introduz dados específicos para testar que todos os caminhos são executados.
- *Statement test:* Detecta se cada uma das linhas da função foi executada no mínimo uma vez.

De acordo com a norma DO-178B, esta análise dinâmica é chamada de análise de cobertura e as técnicas aplicadas são:

- *Cobertura funcional:* Detecta se todas as funções foram chamadas
- *Cobertura de linha:* Cada uma das linhas da função foi executada no mínimo uma vez.
- *Cobertura de decisão:* Detecta se em cada rama de toma de decisão de código todas as decisões foram executadas.
- *Cobertura de condição:* Detecta se em cada rama de condição de código todas as condições foram executadas.
- *Cobertura de decisão e condição:* Detecta se todas as combinações de decisão/condição foram executadas.
- *Cobertura de estados:* Detecta se numa máquina de estados do software, todos os estados foram alcançados uma vez no mínimo.

7.2 Níveis de aplicação das técnicas

O processo de verificação e validação acompanha todas as fases do ciclo de vida de projeto. Usando como guia o modelo em V do ciclo de vida de um projeto representado na figura 1, pode-se apreciar que para cada fase de desenvolvimento (a rama es-

querda da V) os testes devem ser escritos para depois, na rama esquerda, serem aplicados sobre o software já implementado. Com este método todos os requisitos tem teste associado, assegurando que todos eles são implementados e testados.

Assim, o processo de V&V se aplica em todos os níveis de desenvolvimento do produto. Nesta seção estes níveis são apresentados e a utilidade do processo de V&V nesse nível é explicada (ESA (1995)).

- *Unitário*: As técnicas são aplicadas aos componentes unitários isolados. O objetivo é comprovar que o componente responde às especificações (requisitos de baixo nível) e pode ser integrado.
- *Integração*: As unidades são integradas e testadas em conjunto. A integração se faz em múltiplas etapas até ter o sistema completo. O objetivo é comprovar o correto funcionamento das interfaces entre componentes.
- *Sistema*: O propósito neste nível é verificar que o sistema cumpre com as especificações (requisitos de alto nível). São escolhidos cenários típicos de atuação do software para testar o funcionamento global do sistema.
- *Aceitação:* Neste nível deve-se comprovar que o software cumpre os requisitos acordados com o usuário.

7.3 Outras técnicas de V&V

As técnicas de V&V acima detalhadas são as mais conhecidas e utilizadas. Estas são as mais comumente apresentadas nas bibliografias.

Além destas técnicas existem algumas outras também utilizadas mais menos conhecidas e, habitualmente, não tão rigorosamente classificadas. Em seguida, algumas delas são explicadas, vide RPG (2000), para maiores informações:

- *Testes Alpha e Beta:* Este tipo de teste consiste em oferecer a usuários potenciais (internos à firma no caso dos testes alpha e externos a ela no caso dos testes beta) uma versão do software. Estes usuários usarão o aplicativo e reportarão as falhas ao desenvolvedor.
- *Testes de regressão:* Estes testes são aplicados ao software depois de ter solucionado falhas encontradas. O objetivo é comprovar que a solução dessas falhas não introduziu novos erros no software.

8 Conclusão

O objetivo deste artigo é fazer uma revisão do processo de verificação e validação aplicado a software seguro e crítico, definindo os conceitos mais utilizados, enumerando as limitações do processo, determinando os objetivos principais, detalhando algumas normas aplicadas e classificando técnicas e níveis de aplicação delas.

O processo de verificação e validação de software é extremamente importante quando se trata de sistemas críticos e seguros, onde uma falha pode causar perdas humanas.

Segundo a norma MIL-STI-1629A, O software crítico e seguro é classificado em quatro níveis em função da severidade do efeito que uma falha pode causar. Estes níveis são catastrófico, marginal, maior e menor.

Para atingir certo nível de qualidade do software, existem normas que devem ser cumpridas. Assim este software não pode ser comercializado sem antes ser certificado em relação ao cumprimento dessas normas.

As normas aplicadas são diferentes segundo a área de aplicação (DO-178B/C para aeronáutica ou ISO 26262 para automobilística), e dependendo do nível de severidade de software, diferentes documentos devem ser gerados para as autoridades de certificação.

Existem várias técnicas de verificação e validação aplicáveis a diferentes níveis do ciclo de desenvolvimento em função do objetivo do teste.

Referências Bibliográficas

http://www.softwaretestinggenius.com.

James S. Collofello (1988). Introduction to software Verification and Validation.

ESA (1995) Guide to software verification and validation. http://cisas.unipd.it/didactics/STS_school/Software_development/Guide_to_the_SW_verification_and_validation-0510.pdf

MIL-STD-1629A (1980). Procedures for performing a failure mode, effects and criticality analysis.

Haik Biglari (2008). Past, Present and Future of Safety-Critical Real-time Embedded Software Development.

U. Fujii. Software Verification and Validation (V&V)

A. Stellman (2005). Aplied Software Project Management. http://www.stellman-greene.com/aspm/content/blogcategory/32/40/

RPG Reference Document (2000). VV&A RPG Reference Document. V&V Techniques.

M. Mainar (2009). Testing Safety Critical Software Systems. University of Nottingham.

L. Gayarre (2013). An overview of models, methods and tools for verification, validation and accreditation of real time critical software.

