Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitungarbeit

Pascal Behrmann

Software FMEA und Fehlerbaumanalyse

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Pascal Behrmann

Thema der Ausarbeitung

Software Failure Modes and Effects Analysis und Fehlerbaumanalyse

Stichworte

Failure Modes and Effects Analysis, Fault Tree Analysis, Fehlerbaumanalyse

Kurzzusammenfassung

Failure Modes and Effects Analysis (FMEA) und Fault Tree Analysis sind im Bereich der Industrie weit verbreitet. Diese Ausarbeitung verschafft einen Überblick über die Methoden des Software FMEA und beschreibt die Grundlagen einer Anwendbarkeitsuntersuchung von FMEA für die Fehlerpropagation bei objektorientierter Anwendungssoftware.

Pascal Behrmann

Title of the paper

Software Failure Modes and Effects Analysis and Fault Tree Analysis

Keywords

Failure Modes and Effects Analysis, Fault Tree Analysis

Abstract

Failure Modes and Effects Analysis and Fault Tree Analysis are well understood for hardware and in widespread use in the industrial field. This paper supplies the reader with a summary of the ideas and methods of software FMEA and describes the principles of a practicability survey for FMEA and fault propagation for object oriented application software.

Inhaltsverzeichnis

1 Einführung

Die Entwicklung von Software ist ein fehleranfälliger Prozess. Herkömmliche Test- und Analysemethoden überprüfen nur das Verhalten auf erwartete Eingaben, hierbei können erwartete Eingaben auch fehlerhafte Eingaben sein, die jedoch vom Testenden als mögliches Fehleingaben oder Systemausfälle angesehen wurden. Tests können jedoch nicht jeden möglichen Fall abdecken. Failure Modes and Effects Analysis (FMEA) untersucht die Auswirkungen jedes theoretisch vorhandenen Fehlers in einer Komponente. FMEA liefert eine Methode die Auswirkungen von einzelnen Ausfällen oder Fehlerzuständen auf das Systemverhalten analysiert. Somit beantwortet sie die Frage, was der größte Anzunehmende Ausfall des Softwaresystems durch das Versagen einer einzelnen Komponente ist.

FMEA analysiert vorhandene Artefakte der Entwicklung und geht von Fehlerzuständen in einzelnen Modulen aus um die Effekte auf das Gesamtsystem zu analysieren. Hierbei wird betrachtet wie sich Fehlerzustände durch das System ausbreiten, welchen Schweregrad sie haben und dadurch implizit an welchen Stellen Überprüfungen eingebaut werden müssen.

Fault Tree Analysis (FTA) benutzt einen entgegengesetzten Ansatz, für den Fehlerbaum werden Fehlerwirkungen angenommen und ihre Quelle gesucht. Hierdurch ist es möglich die Zustände zu identifizieren die zu schweren Fehlereffekten führen. Im Gegensatz zu FMEA erkennt diese Methode auch Fehlerwirkungen die durch mehrere Ausfälle erzeugt werden.

2 Software Failure Modes and Effects Analysis

Failure Modes and Effects Analysis ist eine analytische Methode zur Feststellung von Fehlereinflüssen. Diese Methode besitzt innerhalb der Industrie einen hohen Verbreitungsgrad. Mehrere Standards sind im Laufe der Zeit veröffentlicht worden, dies begann durch militärische Standards wie den MIL-P-1629A durch die DOD (1980), dem ersten Standard zur FMEA, später wurden jedoch auch industrielle Standards, wie die des AIAG (2008) veröffentlicht.

FMEA für Softwareentwicklungen muss jedoch an die speziellen Eigenheiten von Software angepasst werden. Anders als Maschinen und elektronische Geräte nutzten Softwarefunktionen nicht mit der Zeit ab. Defekte einer Software sind bereits bei der Auslieferung vorhanden, treten jedoch erst unter spezifischen Bedingungen auf. Ebenfalls existieren keine spezifischen Ausfallwahrscheinlichkeiten, die auf eben jener Abnutzung basieren. Da jedoch die Ausfallwahrscheinlichkeit ein wichtiger Indikator für die Relevanz einer Fehlerwirkung ist, fällt ein wichtiges Bewertungskriterium des FMEA Prozesses weg. Damit ist ist SFMEA keine Analyse von Wahrscheinlichkeiten, sondern eine Analyse von Möglichkeiten, jede Auswirkung des Ausfalls jeder Systemkomponente wird betrachtet.

2.1 Grundlagen des Software FMEA

Software FMEA analysiert vorhandene Artefakte des Entwicklungsprozesses um die Wirkungen von Fehlerzuständen zu erhalten. Hierbei wird betrachtet welche Funktionen von einem Fehlerzustand betroffen sind und wie schlimm sie betroffen sind. Grundsätzlich wird dabei angenommen, dass jede Funktion in einen Fehlerzustand geraten kann, eine Variable oder ein Eingabe einen falschen oder ungewollten Wert beinhalten kann und die Propagierung dieses fehlerhaften Wertes auf andere Variablen wird analysiert. Melinda Kennedy, nach Pentti und Atte (2002), beschreibt den Zweck von FMEA als:

- Identifizierung von designbedingten Fehlerzuständen, die durch eine Änderung des Designs entfernt werden können.

- Die Effekte von Fehlerzuständen auf die Systemfunktionalität herauszufinden.
- Den Grund für Fehlerzustände herauszufinden.
- Priorisierung von empfohlenen Gegenmaßnahmen zu bestimmten Fehlerzuständen durch Betrachtung der schwere der Fehlerauswirkung und der Wahrscheinlichkeit des Auftretens.
- Identifizierung, Implementation und Dokumentation der Gegenmaßnahmen.

Somit ermöglicht ein korrekt durchgeführtes Software FMEA das Auflisten von möglichen Gegenmaßnahmen, mit denen das Auftreten von kritischen Fehlern verhindert werden kann. Aus diesem Grund wird Software FMEA für Projekte im Bereich der sicherheitskritischen Softwareentwicklung für Embedded Systems verwendet.

Software Failure Mode, Effects, and Criticality Analysis

Software Failure Mode, Effects, and Criticality Analysis (SFMECA) ist eine Erweiterung des Software Failure Modes and Effects Analysis die Kritikalität und Feststellbarkeit von Fehlerzuständen betrachtet. Hierzu wird jedem Fehlerzustand bei seiner Betrachtung ein Wert zwischen 1-10 zugewiesen, wobei 10 sowohl den höchsten Schweregrad als auch die geringste Feststellbarkeit eines Fehlers bedeuten, so dass alle Fehlerzustände nach dem Produkt der beiden Zahlen geordnet werden können.

2.2 Prozedur des SFMEA

Aufgrund fehlender Standards für Software FMEA existiert keine allgemeingültige Prozedur, die sich in der Literatur finden lässt. Da die Unterschiede der Vorgehensweisen der dieser Ausarbeitung zugrundeliegenden Quellen vor allem in der Ausführungsreihenfolge, der Aufteilung der Aufgaben und dem hinzufügen von weiteren Schritten niederschlägt, lässt sich jedoch ein Vorgehensmodell reduzieren. Das im folgenden beschriebene Vorgehen basiert auf dem FMEA Standard 60812 des IEC (2006), wie von Pentti und Atte (2002) beschrieben, sowie der von Ozarin (2008) vorgeschlagenen Prozedur.

Systemgrenzen der Analyse bestimmen

In dieser Phase wird bestimmt welche Teile des Systems untersucht werden müssen und können. Diese Grenzen hängen von dem momentanen Stand der Entwicklung, der Größe des zu entwickelnden Systems und den Anforderungen an das Ergebnis der Analyse ab.

In dieser Phase kann bestimmt werden auf welcher ebene die FMEA stattfinden wird. Dies kann auf der Ebene des Modells, des Codes der Software oder einer dazwischenliegenden Ebene durchgeführt werden.

Codelevel Analyse liefert genauere Ergebnisse als Analysen des Modells, kann sich jedoch als unpraktisch erweisen, da der Aufwand für die Analyse des Codes höher ist, oder diese gar unmöglich ist, da in frühen Entwicklungsphasen der Quellcode noch nicht, oder noch nicht vollständig, zur Verfügung stehen.

Funktionen und Anforderungen des Systems verstehen

Die Anforderungen an die Funktionalität und Sicherheit eines Systems, sowie seine exakte Funktion zu verstehen, ist der wichtigste, jedoch auch ein hochkomplexer, Schritt. Da FMEA, wie jede Methode zur Unterstützung der Softwareentwicklung, möglichst früh beginnen sollte werden zum Zeitpunkt des Analysebeginns oft nur wenige Artefakte der Softwareentwicklung vorhanden sein. Ein Verständnis für diese Anforderungen ist jedoch unumgänglich für die weiteren Schritte.

Kriterien für Fehlschlag und Erfolg festlegen

Um die Auswirkungen auf das Systemverhalten von Fehlerwirkungen einzuschätzen ist es notwendig zu Wissen welche Funktionen zum Erfolg oder Fehlschlag der Systemfunktion beitragen.

System in Teile zerlegen

Durch die Modularisierung von Softwaresystemen sollten bei einem guten Systementwurf keine Probleme bei der Zerlegung des Softwaresystems auftauchen. Abhängig von der Ebene der Betrachtung sind die kleinsten Einheiten die von der FMEA untersucht werden. Untersuchungen auf der Ebene des Systems oder des Modells arbeiten mit Klassen und Funktionsaufrufen, während Analysen des Quelltextes auf Variablen und einzelnen Konstrukten der Programmiersprache basieren.

Fehlerzustände und Fehlerwirkung

Sowohl auf Code Ebene, als auch auf einer höheren Ebene, können für jedes zu betrachtende Element, dies können wie erwähnt in Abhängigkeit von der Ebene beispielsweise Variablen, Instruktionen, Funktionen oder Pakete sein, mehrere mögliche Fehlerzustände. Für die Ebene des Codes sind diese Zustände gut Verstanden und können somit wiederverwendet werden. Für höhere Ebene hängen diese Fehlerzustände von der fachlichen Bedeutung des fehlgeschlagenen Elements ab. Eine Fehlerwirkung nach Außen tritt auf, sobald ein Element fehlschlägt, das vom Benutzer oder einem Fremdsystem aufgerufen wurde.

Bestimmung von Fehlerwirkung einzelner Fehlerzustände

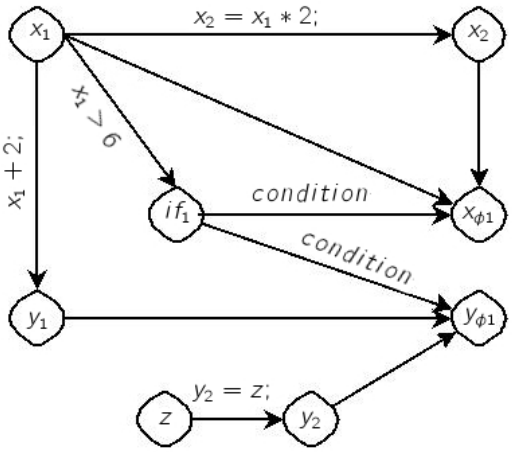
In diesem Schritt wird für jedes betrachtete Element, einzeln, jeder mögliche Fehlerzustand angenommen und die daraus folgenden Fehlerzustände bestimmt. Dieser Fehler wird nun durch das System propagiert, hierzu kann ein Graphensystem wie es von Price und Snooke (2008) beschrieben wurde, zu Hilfe genommen werden. Auf der Ebene des Quellcodes stellen Statements die Kanten und Variablen die Knoten des Graphen dar, auf einer höheren Ebene sind zum Beispiel dies Funktionsaufrufe und Objekte.

Im folgenden wird die Fehlerpropagation anhand eines kleineren Beispiels auf Code Ebene erläutert, weiterführende Quellen lassen sich im Literaturverzeichnis ab Seite 16 finden.

Beispiel einer Fehlerpropagation

Für die Propagierung von Fehlerzuständen auf andere Variablen ist es notwendig den Quellcode in eine andere Darstellung umzuwandeln. In diesem Beispiel wird die Propagierung von Fehlerzuständen auf Code Ebene unter Zuhilfenahme der Static Single Assignment (SSA) Darstellung und eines Fehlerpropagierungsgraphen beschrieben. Hierbei wird bei jeder Zuweisung der zugewiesenen Variable ein neuer Index zugeteilt. Dies hat zur Folge das jeder Variable nur einmal ein Wert zugewiesen wird. Ein Fehlerpropagierungsgraph ist eine graphische Darstellung der SSA Form, in der, wie bereits erwähnt, Variablen als Knoten und Zuweisungen als Kanten abgebildet werden.

Um eine Bedingte Anweisung zu realisieren wurde eine Orakelfunktion(ϕ), wie sie nach Price und Snooke (2008) von Collard genutzt wurde, eingeführt. Diese Funktion wählt aus verschiedenen Variablen eine Variable aus, um den nicht-linearen Programmfluss darzustellen.

Quellcode	SSA Darstellung	Graph
$x = 1;$ $y = x + 2;$ $if(x_1 > 6)\{$ $y = z;$ $x = x * 2;\}$	$x_1 = 1;$ $y_1 = x_1 + 2;$ $if_1 = x_1 > 6$ $condition\{if_1$ $y_2 = z;$ $x_2 = x_1 * 2;\}x_{\phi 1} = \phi(x_1, x_2)$ $y_{\phi 1} = \phi(y_1, y_2)$	

Aus dem Graphen ist es einfach zu erkennen, dass Fehlerzustände sich auf y propagieren, sich jedoch Fehlerzustände von x nicht auf y propagieren, da es keine Graphenverbindung von y zu x gibt. Unter Zuhilfenahme von Graphenalgorithmen lassen sich auf diese Art und Weise auch komplexere Graphen auf ihre Eigenschaften untersuchen. Falls in einem späteren Teil des Programmcodes mit x und y Funktionen aufgerufen werden, wird sich der Fehlerzustand, bei einem fehlerhaften x-Wert damit sowohl bei Aufrufen mit x als auch bei aufrufen mit y fortpflanzen. Bei einem fehlerhaften y-Wert sind nur Funktionsaufrufe mit y betroffen.

Erstellen des Berichtes

Ein Analysebericht wird erstellt um Entwickler über die Hauptfunde der FMEA, grundsätzliche Annahmen und die Bedeutung der Fehlerzustände und Fehlerwirkungen zu informieren. Jene Elemente und Fehlerzustände, die die meisten Fehlerwirkungen erzeugen werden aufgelistet, damit diese von den Entwicklern abgesichert werden können. Zusätzlich kann ein solcher Bericht Fault Trees enthalten, die im Kapitel 3 vorgestellt werden. Ein Beispiel für ein SFMEA Formular befindet sich in Anhang A.1.

2.3 Probleme des SFMEA

Die grundlegenden Probleme des Software FMEA liegen vor allem in der Größe des Prozesses. Für ein vollständiges Software FMEA auf Code-Ebene ist es notwendig jede Zeile Code mehrfach durchzugehen, da jede Funktion mehrere mögliche Fehlerzustände hat, deren Auswirkungen einzeln analysiert werden müssen. Wenige Zeilen Code können auf diese Art und Weise, bereits Seiten mit möglichen Fehlerwirkungen füllen. Große Programme erzeugen somit tausende Seiten, die mit Fehlerwirkungen und Fehlerzuständen gefüllt sind.

Ein weiteres Problem entsteht dadurch, dass FMEA nur einzelne Fehlerzustände betrachtet, nicht jedoch die Auswirkung mehrerer gleichzeitig auftretender Fehlerzustände. Eine FMEA Untersuchung kann damit ergeben, dass eine Software vollständig Fehlerfrei ist, in der Realität kann diese trotzdem noch immer Fehlerwirkung zeigen.

2.4 Automatisierung von SFMEA

Wie zuvor beschrieben ist das große Problem der Software FMEA ihr großer Aufwand. Eine Automatisierung dieses Prozesses ist durch seine einfache und sich wiederholende Struktur möglich und wurde bereits in verschiedenen Projekten durchgeführt, wie von Ozarin und Siracusa (2003) und Price und Snooke (2008) beschrieben wurde.

Durch die Automatisierung dieses Teils des FMEA Prozesses wird jedoch nicht das Problem beeinflusst, dass enorme Mengen an Daten generiert werden, die nur schwer zu Handhaben sind. Laut Pentti und Atte (2002) sind FMEA Arbeitsblätter mindestens 16 Zeilen breit und manchmal einige tausend Seiten lang. Die Verarbeitung dieser Informationsmasse muss, zur effizienten Nutzung des Verfahrens, also ebenfalls durch Automatismen unterstützt werden. Price und Snooke (2008) beschreiben die Entwicklung einer nicht sicherheitskritischen Anwendung unter Zuhilfenahme von FMEA, in der es durch Automatismen möglich war die Datenmengen auf wichtige Informationen zu reduzieren.

3 Fehlerbaumanalyse

Fehlerbaumanalyse (FTA) stellt die Gegenrichtung des FMEA dar, hierbei wird eine Fehlerwirkung als gegeben angenommen und es werden alle Quellen, unter Zuhilfenahme der booleschen Algebra, aufgelistet. Dies geschieht indem von der Fehlerwirkung, als Wurzel ein Baum aufgebaut wird, der die jeweiligen Quellen der Fehlerzustände aufzeigt, die die Fehlerwirkungen hervorgerufen haben. Dies wird durchgeführt bis die grundlegenden Ursachen für die Fehlerwirkung gefunden sind. Fehlerbaumanalyse ist im Gegensatz zu FMEA eine top-down Analysemethode und ist ein integraler Bestandteil der Sicherheitsanalyse. Da für das geplante Projekt diese Analysen jedoch von untergeordneter Relevanz sind, wird auf die Ideen und Techniken dieser Methodik nur der Vollständigkeit halber eingegangen.

3.1 Grundlagen

Die Elemente der Fehlerbaumanalyse bestehen aus Ereignissen und Gattern. Ereignisse sind Wirkungen von Elementen und können sich in weitere Bäume aufspalten, ein Ereignis das keinerlei weitere Aufteilung mehr ermöglicht wird als Primärereignis bezeichnet. Es existieren verschiedene Arten von Gattern, die im folgenden in Kürze vorgestellt werden.

AND: Das Folgeereignis geschieht, wenn alle eingehenden Ereignisse geschehen sind.

OR: Das Folgeereignis geschieht, wenn ein eingehendes Ereignis geschehen ist.

KOFM: Das Folgeereignis geschieht, wenn mindestens k von m eingehenden Ereignissen geschehen sind.

Priority AND (PAND): Das Folgeereignis geschieht, wenn alle eingehenden Ereignisse in der selben Reihenfolge geschehen sind, in der sie als Eingehend dargestellt sind.

Es existieren noch weitere Gatter, die Ausfallsicherungen und Reserven darstellen, bestimmte Reihenfolgen von Ereignissen erzwingen oder funktionale Abhängigkeiten ausdrücken können, diese werden hier jedoch nicht behandelt.

3.2 Prozedur

Das Vorgehen während einer Fehlerbaumanalyse ähnelt dem beschriebenen Vorgehen der FMEA.

Auswahl der zu betrachtende Fehlerwirkung: Die Auswahl der relevanten Fehlerwirkung, die während einer Fehlerbaumanalyse betrachtet wird kann ein hohes Maß an Systemkentniss erfodern. Während einige Fehlerwirkungen offensichtlich sind, können andere kritische Fehlerwirkungen erst mit einem Verständnis für das System erkennbar werden.

Fehlerbaum konstruieren: Sobald die Fehlerwirkung ausgewählt wurde, werden in einem Verfahren, das der FMEA ähnelt, die möglichen Auslöser einer Fehlerwirkung bestimmt. Anders als bei der FMEA ist es mit dieser Analysemöglichkeit möglich auszudrücken, dass mehrere Auslöser für eine Fehlerwirkung verantwortlich sind.

Fehlerbaum evaluieren: Nach der Erstellung wird der Fehlerbaum auf mögliche Änderungen des Softwaredesign untersucht, die verhindern würden, dass der unerwünschte Effekt auftritt.

Erstellen des Berichtes: Ähnlich wie im Bericht der FMEA besteht der Sinn des hier erstellten Berichts daraus, die Entwickler des untersuchten Softwaresystems über mögliche Maßnahmen zu informieren.

3.3 Probleme

Wie bereits bei der FMEA ist das größte Problem der Fehlerbaumanalyse die Größe des Fehlerbaums. Die Wachstumsrate des Fehlerbaums ist jedoch so groß, dass es praktisch kaum möglich ist den vollständigen Fehlerbaum eines Softwaresystems zu überblicken. Aus diesem Grund wird ein Fehlerbaum in der Praktischen Anwendung nur bis zu einer gewissen Tiefe erzeugt und die Ergebnisse der FMEA Untersuchung werden komplementär genutzt. Hierbei werden die in der FMEA erzeugten Ergebnisse als Blätter des Fehlerbaums genutzt.

3.4 Automatisierung

Wie in Kapitel 2.2 gezeigt, ist es möglich einen Quellcode in einen Graphen umzuwandeln. Die Fehlerwirkung einer spezifischen Komponente zu ihren Auslösern zu verfolgen ist durch die Nutzung dieser Graphen ein automatisierbares Problem. Wie bereits bei der FMEA Analyse ist hierdurch jedoch nur die Konstruktion eines Graphen automatisiert, die Evaluierung

und das Verstehen des Systems können durch automatische Systeme nur unterstützt werden.

4 Ausblick

Software Failure Modes and Effects Analysis ist eine ausgezeichnete Ergänzung zu anderen Analysemethoden und Tests. Die erwähnten Ergebnisse von Price und Snooke (2008) zeigen, dass es möglich ist große Teile des SFMEA Prozesses zu automatisieren. Wie zuvor erwähnt erlaubt die Sichtweise der SFMEA es auf verschiedenen Ebenen Analysen durchzuführen, damit ist es möglich die Analyse bereits in der Designphase der Softwareentwicklung zu starten und über den vollständigen Entwicklungsprozess zu begleiten. Der Kosten-Nutzen-Faktor für ein nicht sicherheitskritisches Projekt lässt sich nur schwer abschätzen. Eine SFMEA auf Codeebene wurde von Price und Snooke (2008) an einem Reisekostenrückerstattungssystem automatisiert durchgeführt. Das Ergebnis dieser Analyse schien positiv für SFMEA bei Anwendungssoftware zu sein.

4.1 Nutzbarkeitsanalyse von SFMEA für objektorientierte Anwendungssoftware anhand einer mobilen Spieleanwendung

Software Failure Modes and Effects Analysis für sicherheitskritische Systeme ist ausreichend Untersucht und der Nutzen dieser Analyseform ist bereits von Experten in verschiedenen Projekten gezeigt worden. Artikel von Hecht u. a. (2004) und Price und Snooke (2008) haben die Frage aufgeworfen ob diese Analyseform auch für objektorientierte Anwendungsentwicklung genutzt werden kann. Die Unterstützung eines Softwareentwicklungsprozesses zur Implementierung einer mobilen Spieleanwendung im Bereich des Pervasive Gaming soll zeigen, ob eine solche Analyse für diese Art der Softwareentwicklung geeignet ist. Dabei sollen möglichst viele Artefakte, die während der Softwareentwicklung erzeugt werden, genutzt werden um Aussagen zu treffen. Diese Aussagen werden vor allem die Propagierung von Fehlern betreffen, da sicherheitskritische Aspekte für diese Art der Anwendung weniger Relevanz haben.

Zum Zweck der Analyse wird ein Framework implementiert werden müssen, das diese soweit wie möglich automatisiert, da der Aufwand einer solchen Analyse eine essentielle Frage des Projektes darstellt. Weiterhin muss der Nutzen der SFMEA Analyse erforscht werden, hierzu

werden die gefundenen Probleme, die dazugehörigen Lösungswege und ob sie bereits ohne die SFMEA offensichtlich waren.

Die Methodik der SFMEA könnte erweitert werden, um Fehlerzustände in mehreren Elementen zugleich anzunehmen, wo diese Wahrscheinlich zugleich auftreten. Eingaben die von einem Teil des Systems, zum Beispiel einem der mobilen Clients, oder von einem Benutzer stammen könnten gehäuft felerhaft auftreten, wenn der Benutzer einen Fehler in der Eingabe macht, oder Daten falsch übertragen werden. So ließe sich auch der größtmögliche Schaden durch mutwilliges Verhalten aufspüren.

Um die Korrektheit der Automatisierung zu zeigen ist es notwendig zumindest Beispielhaft das Framework an einem sicherheitskritischen System zu testen. Hierzu sollte ein Literaturbeispiel, das ausreichend beschrieben und verstanden wurde herangezogen werden. Dadurch ist es möglich die Probleme des getesteten System den vom Framework gefundenen Problemen gegenüberzustellen.

Risiken

Ein unabsehbares Risiko ist, dass die Analyse der Anwendungssoftware fehlschlägt und keine nutzbaren Ergebnisse liefert. In diesem Fall wäre es notwendig zu zeigen, dass das Scheitern aufgrund eines grundsätzlichen Problems des Ansatzes der SFMEA für Anwendungssoftware ist und nicht ein Problem der Anwendung der Analyse. Dies könnte sich jedoch als äußert schwierig oder unmöglich erweisen. Falls der Analyst die Prozedur der SFMEA falsch oder unzureichend anwendet, oder die implementierten Automatismen nicht ausreichen, obwohl bessere Automatismen möglich wären und dadurch ein Fehlschlag der Analyse festgelegt wird, wäre das Projekt ein Fehlschlag, da diese Festlegung inkorrekt wäre.

Weitere Risiken ergeben sich aus dem Softwareprojekt, falls dieses fehlschlägt, oder nicht ausgeführt wird, ist es, unter Umständen, nicht möglich die oben erwähnten Aussagen zu sammeln und eine Aussage über SFMEA in Bezug auf objektorientierte Anwendungssoftware zu treffen. In einem solchen Fall muss ein Alternativprojekt gefunden werden, dass diese Aufgabe übernimmt.

Literaturverzeichnis

[AIAG 2008] AIAG, Automotive Industry Action G.: *Potential Failure Mode and Effect Analysis (FMEA), 2th Edition*. 2008

[DOD 1980] DOD: *Military Standard Procedures for Performing a Failure Mode, Effects and Criticality Analysis*. 1980

[Hecht u. a. 2004] HECHT, Herbert ; AN, Xuegao ; HECHT, Myron: Computer Aided Software FMEA for Unified Modeling Language Based Software. In: *RAMS* (2004), S. 243–248

[IEC 2006] IEC: *Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA)*. Januar 2006

[Ozarin 2008] OZARIN, Nathaniel: The Role of Software Failure Modes and Effects Analysis for Interfaces in Safety- and Mission-Critical Systems. In: *Proceedings of the 2nd Annual Systems Conference* (2008), April

[Ozarin und Siracusa 2003] OZARIN, Nathaniel ; SIRACUSA, Michael: A Process for Failure Modes and Effects Analysis of Computer Software. In: *Proceedings of the Annual Reliability and Maintainability Symposium* (2003), S. 365–370

[Pentti und Atte 2002] PENTTI, Haapanen ; ATTE, Helminen: Failure Mode and Effects Analysis of Software-Based Automation Systems. In: *STUK-YTO-TR 190* (2002), August

[Price und Snooke 2008] PRICE, Chris ; SNOOKE, Neal: An Automated Software FMEA. In: *Proceedings of the International System Safety Regional Conference, Singapore* (2008), April

A Formulare

A.1 SFMEA Formular

Ref-No	Component	Fault	Cause	Failure Effect	Safety

Glossar

Anwendungssoftware Software ohne sicherheitskritischen Hintergrund.

Artefakt Ein Ergebnis des Softwareentwicklungsprozesses, zum Beispiel Quellcode, eine Anforderungsanalyse oder ein Modell des Systems.

Failure Mode, Effects, and Criticality Analysis Eine Erweiterung der FMEA Methodologie, die die Kritikalität eines Fehlermodus betrachtet.

Failure Modes and Effects Analysis Eine Methodik, die Auswirkungen von Fehlerzuständen analysiert.

Fault Tree Analysis Eine Top Down Analyse, die Fehlerzustände findet, die für bestimmte Fehlerwirkungen verantwortlich sind.

Fehlerwirkung Auswirkung eines Fehlereffekts auf das Verhalten des Systems.

Fehlerzustand Fehlerhafter Zustand einer Komponente des Systems.

FMEA Failure Modes and Effects Analysis

FMECA Failure Mode, Effects, and Criticality Analysis

FTA Fault Tree Analysis

SFMEA Failure Modes and Effects Analysis für Softwaresysteme

SFMECA Failure Mode, Effects, and Criticality Analysis für Softwaresysteme