Forensic Science International 259 (2016) 234–245

Contents lists available at ScienceDirect

Forensic Science International

journal homepage: www.elsevier.com/locate/forsciint

A near-miss management system architecture for the forensic investigation of software failures

M.A. Bihina Bella [*], J.H.P. Eloff

ICSA Research Lab, Computer Science Department, University of Pretoria, Pretoria, South Africa

ARTICLE INFO

Article history:
Received 28 June 2015
Received in revised form 2 October 2015
Accepted 7 October 2015
Available online 28 October 2015

Keywords:
Software failure
Near miss
Near-miss management system (NMS)
Digital evidence
Digital forensics

ABSTRACT

Digital forensics has been proposed as a methodology for doing root-cause analysis of major software failures for quite a while. Despite this, similar software failures still occur repeatedly. A reason for this is the difficulty of obtaining detailed evidence of software failures. Acquiring such evidence can be challenging, as the relevant data may be lost or corrupt following a software system's crash. This paper proposes the use of near-miss analysis to improve on the collection of evidence for software failures. Near-miss analysis is an incident investigation technique that detects and subsequently analyses indicators of failures. The results of a near-miss analysis investigation are then used to detect an upcoming failure before the failure unfolds. The detection of these indicators – known as near misses – therefore provides an opportunity to proactively collect relevant data that can be used as digital evidence, pertaining to software failures. A Near Miss Management System (NMS) architecture for the forensic investigation of software failures is proposed. The viability of the proposed architecture is demonstrated through a prototype.

1. Introduction

According to Laprie [1] "a system failure occurs when the delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service". Therefore, for the purposes of the research in hand, a software failure is defined as the unplanned cessation of a software system to function as specified. Software systems can fail for various reasons including system overload, logic errors, security breaches, human errors, and glitches in routine maintenance operations (e.g. failed software upgrade) [2]. As software is embedded in a range of devices and plays a vital role in a number of industries, a failed software application can affect any area of a user's day-to-day life and may even be fatal.

Consider for example the various cases of software failures in medical devices such as radiation therapy machines, external infusion pumps and implantable pace makers. In radiaton therapy machines in particular, failures of the embedded software system causes serious problems such as overdosage of radiation and administration of incorrect treatment that result in severe burns or deaths of the affected patients. Such catastrophic cases of radiation therapy software glitches have been reported many times in the media [3] as well as on the portals of the FDA (the U.S. Food and Drug Administration) [4] and the IAEA (International Atomic Energy Agency) [5].

Disastrous events as the above mentioned often result in lawsuits where a thorough post-mortem investigation is conducted. Comprehensive forensic reports are available for these cases, but do not address the software aspect of the investigation. Such an investigation is absolutely necessary to prevent the recurrence of these catastrophes. To this end, a digital forensic investigation is required to understand the root causes involved in the software failures.

Digital forensics is the process of methodically examining computer media as well as network components, software and memory for digital evidence [6]. This evidence is usually in the form of system logs, but may include other relevant data such as digital images. The digital evidence is used to provide clarity on the cause and circumstances of a computer-based event in support of the criminal justice system. As such, digital forensics is primarily used for the investigation of computer crimes and security-related events (e.g. breach of company policy). Nevertheless, we argue that it can also be applied to non-criminal events such as catastrophic failures that require a court case as the examples provided earlier. In such cases, using digital forensics instead of existing informal

* Corresponding author. Tel.: +27 731497384.
E-mail addresses: mbihina@yahoo.fr (M.A. Bihina Bella), eloff@cs.up.ac.za (J.H.P. Eloff).

http://dx.doi.org/10.1016/j.forsciint.2015.10.007
0379-0738/© 2015 Elsevier Ireland Ltd. All rights reserved.

failure analysis techniques has the benefits of providing results admissible in a court of law due to the scientific foundation and the sound digital evidence used for the root-cause analysis.

However, being a reactive process, digital forensics can only be applied after the occurrence of a failure. This limits its effectiveness as data that could serve as potential evidence may be destroyed during and after the failure. Acquiring such data is necessary for the validity of the results of the forensic investigation. The international ISO/IEC 27037 standard – Guidelines for the Identification, Collection, Acquisition and Preservation of Digital Evidence [7] – indeed recommends that the evidence collection should be prioritised based on volatility.

In order to address this limitation of digital forensics, it is suggested that evidence collection be started at an earlier stage, *before* the software failure actually unfolds, so as to detect the high-risk conditions that can lead to a major failure. These high-risk conditions, so-called forerunners to failures, are known as near misses. By definition, a near miss is a high-risk event that could have led to an accident, but did not, due to some timely intervention or by chance [8]. Almost all major accidents are preceded by a number of near misses [9]. Contrary to other precursors to the failure, a near miss is the closest to the point of failure; in other words, it is the closest to the time window during which the failure occurs. This concept can be better explained with the following example.

Consider for instance a potential car collision at a busy intersection. This potential accident could have been preceded by the following sequence of events: (1) a driver crossing a red traffic light; (2) the driver overspeeding; and (3) the driver struggling to slow down when noticing an incoming car. In the above scenario, the last high-risk event, Event (3), is the near-miss event as it is the closest to the potential crash. The fact that the collision was avoided, maybe due to the carefulness of the driver of the incoming car, makes this sequence of events a near miss.

Near-miss analysis, which refers to the detection and subsequent analysis of near misses, is a technique used in the domain of risk analysis and safety. Like a forensic investigation, near-miss analysis attempts to identify the root cause of accidents and prevent their recurrence and has been used successfully in various industries for decades [9]. It is suggested in this paper that this technique should also be applied to the forensic investigation of software failures. Reason being that the output of a near-miss analysis investigation can be used as digital evidence. Furthermore, it broadens the scope of a forensic investigation so as to also prevent the recurrence of similar software failures. Indeed, as near misses point to the possibly last indicator of an impending failure, they provide a fairly complete set of data about that failure. By alerting system users of an upcoming failure, an opportunity is provided to collect this data at runtime and potentially prevent the failure from unfolding.

Near-miss analysis is usually performed through electronic near-miss management systems (NMS). An NMS that combines near-miss analysis and digital forensics can contribute significantly to the improvement of the accuracy of the failure analysis. However, such a system is not available yet and its design still presents several challenges, due to the fact that neither digital forensics nor near-miss analysis is currently used to investigate software failures and their existing methodologies and processes are not directly applicable to that task.

Preliminary partial solutions to these challenges were presented in Bihina Bella et al. [10] and Bihina Bella et al. [11], respectively, for digital forensics and near-miss analysis. An initial near-miss management model based on these solutions was presented as work-in-progress in Bihina Bella et al. [12]. The current paper presents the revised model and original NMS architecture that resulted from this previous work.

2. Overview of NMS

This section provides some background information on NMSs. It first presents the types of NMSs currently available and then reviews their functionality.

2.1. Types of NMSs

There are essentially two types of NMSs: single or dual. A single NMS only handles near misses, while a dual NMS handles both near misses and accidents [13]. A review of the literature on NMSs indicated that most of the research on near-miss analysis focuses on single NMSs.

Initially limited to the nuclear [9] and aviation industries [14], research on the design of effective NMSs has received much attention in a wide range of industries over the last couple of years [15–18], especially in the healthcare industry for improved patient safety [19–22]. Most NMSs in use today are proprietary systems designed specifically for the organisation that uses them. Barach and Small [19] provide a comprehensive list of proprietary NMSs in various industries.

Apart from proprietary "private" NMSs, some commercial NMSs are publicly available on the market. Commercial NMSs are mostly industry-specific. Examples include AlmostME, an NMS for the medical field [23], and Dynamic Risk Predictor Suite [24], a comprehensive NMS designed for manufacturing facilities.

2.2. Functionality of NMSs

An ideal NMS is required to perform all activities pertaining to near-miss analysis. These activities are summarised in the following diagram in Fig. 1 by Phimister et al. [13]. The diagram uses the following notation:

Dissem: shortcut for dissemination of information
R.C.A: Root-cause analysis
Sol. I.D.: Solution identification

However, most importantly, an NMS focuses on and performs the following three tasks:

- Identification of near misses
- Selection and prioritisation of near misses for analysis
- Root-cause analysis of the selected near misses

2.2.1. Techniques for the identification of near misses

The identification of near misses is often done manually by means of observation. Recognising an observed event or condition as a near miss requires a clear definition of what constitutes a near miss with various supporting examples. Organisations therefore spend considerable effort to formulate a simple and all-encompassing definition of near misses that is relevant for their respective business operations [25,26]. This definition can differ significantly from one industry to the next.

For instance, in the medical field, a near miss is defined as "an event that could have resulted in unwanted consequences, but did

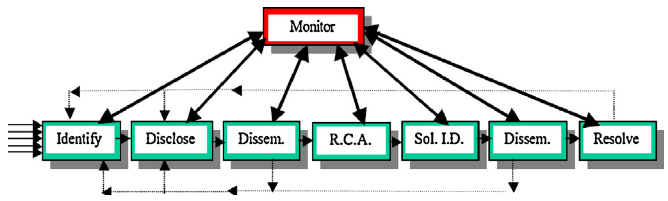

Fig. 1. Near-miss management process [13].