Chapter 3
Near-Miss Analysis: An Overview

3.1 Introduction

During the operations and maintenance phase of a software system, software-related failures can be ascribed to many reasons. Consider, for example, a software system that crash as a result of insufficient cache memory. In this case, during the normal running of the software system, no evidence is collected about the usage and potentially degradation of cache memory. The software system just comes to an unexpected halt without any explanation of what happened. Software systems developed in C++ are a prime example of this type of software failure. Volatile data, such as the number of read and write operations made to external memory while the software system is running, that could serve as potential evidence may be lost following a software failure and may hamper the accurate root-cause analysis of the software failure. Near misses, as immediate precursors to major failures, can address the above issue by providing complete and relevant evidence of the failure before it unfolds. Near-miss analysis, which is originally an accident investigation technique used in a number of high-risk industries, is not yet formally in use in the software industry. Although the learning opportunity offered by the analysis of near misses is not used in the IT industry, its application to the investigation of accidents in other industries is not a new concept at all. The emergence of formal near-miss analysis as currently conducted in a number of scientific disciplines can be traced back to the mid-1970s in the United States. As a near miss is a type of accident sequence precursors (ASP), the history of near-miss analysis is intertwined with the history of ASP analysis. The analysis of ASPs emerged from the need to improve safety in industries that are prone to catastrophic industrial accidents. As is the case with many safety improvement initiatives, ASP analysis was initiated formally following tragic events – in particular two accidents that occurred in the United States in the 1970s (NASA 2006). In both cases, the investigation that followed found some leading events and conditions that could have prevented the disasters had they been identified timely and handled appropriately (Phimister et al. 2004). Hence,

J. Eloff, M. Bihina Bella, *Software Failure Investigation*,
DOI 10.1007/978-3-319-61334-5_3

organisational programs, systems and methodologies were created to detect these precursors and learn from them. The events in question are the following:

- The crash of the American TWA Flight 514 that killed all 85 passengers and seven crew members in 1974 (NASA 2006). Following the investigation of this accident, NASA established the Aviation Safety Reporting System in 1976 to report and analyse observed ASPs in the aerospace industry (NASA 2006).
- The nuclear accident at Three Mile Island that caused the release of toxic gas into the environment in March 1979 (Minarick and Kukielka 1982). Shortly after that accident, the Nuclear Regulatory Commission (NRC) initiated an ASP program to identify, analyse and document ASPs (including near misses) (Phimister et al. 2004).

NASA and the NRC established quantitative and qualitative analysis techniques to determine the risk of a severe accident, based on operational data of observed unsafe events (Belles et al. 2000; Kirwan et al. 2007; NASA 2011). Examples of such events included the degradation of plant conditions and failures of safety equipment (Belles et al. 2000).

The ASP methodology was subsequently adapted for use with other types of industrial accidents and adopted by the respective industries. The latter included the chemical industry (Ritwik 2002; Phimister et al. 2003), the oil and gas industry (Cooke et al. 2011; Vinnem et al. 2010; Skogdalen and Vinnem 2011), the health-care industry (Barach and Small 2000; Sujan 2012) and the finance industry (Mürmann and Oktem 2002). To provide a complete history of ASP analysis falls beyond the scope of this book. However, a fairly comprehensive summary was written by Jones et al. (1999).

Nowadays, the analysis of ASPs and near misses has spread to a wide range of subjects. Saleh et al. (2013) indicate that over 58,000 articles listed in the Web of Science database have the term “precursor” in their title, and this concept is used by around hundred different fields of science. A number of these articles also have the term “near miss” in the title or as a keyword. A keyword search for the term “near miss” in the ScienceDirect database results in over 83,000 articles.

In addition, two major research projects on near-miss analysis provide a significant number of papers on the topic. The first one is the near-miss project at the Risk Management and Decision Processes Centre at the Wharton School, University of Pennsylvania, which has been ongoing since 2000 (Phimister et al. 2000). The researchers conducted more than 100 interviews in several plants in five Fortune 500 companies to assess the near-miss programs managed by their environmental, health and safety departments. The second project is the Accident Precursor Project which was conducted in 2003 by the US National Academy of Engineering. The report of the resulting workshop that extensively reviewed near-miss analysis across industries to promote cross-industry knowledge sharing is available in an online book (Phimister et al. 2004).

Additionally, several study reports have been produced by the ASP program of the NRC (Belles et al. 2000). NASA also published a handbook on precursor analysis in 2011 (NASA 2011). Other research work has been published in workshop proceedings (Bier 1998; Van der Schaaf 1991).

3.2 Background and Review on Near-Miss Analysis

Examples of near misses in software systems can be recognised from the cases of major software failures. One such example is the 3-day BlackBerry outage that occurred in 2011. Other examples are medical accidents involving radiation therapy machines. With regard to the BlackBerry case, the outage was intermittent and comprised a succession of smaller failures, which indicates that different things went wrong at different times. Press reports indicate that firstly, a central server went down, then the backup system failed; e-mail traffic was then rerouted to another main server, which soon became overloaded (Reardon 2011). Each of these unsafe events was a precursor to the subsequent outage. This catastrophic event could have been a mere near miss if the faulty backup server had been replaced or repaired before the second main server became overloaded. Regarding the radiation therapy accidents, the majority of cases were preceded by a number of harmless malfunctions following similar patterns as the deadly accidents. The correct handling of these near misses could have helped prevent the accidental death of a number of cancer patients.

As near misses are a special type of accident precursor, near-miss analysis is a specialised area of the broader field of accident precursor analysis. The American National Aeronautics and Space Administration (NASA), which was the first institution to formally investigate accident sequence precursors, defines accident precursor analysis as "the process by which an organization evaluates observed anomalies and determines if the mechanism at the origin of that anomaly could recur with more severe results" (NASA 2006). This definition refers to accident precursors as "observed anomalies", which is not suitable for the software industry since accident precursors and near misses in software systems may not be visible at all in the absence of a failure.

No standard definition for near-miss analysis, which is also referred to as near-miss management, is available in the literature. Authors on this topic usually focus on defining the near-miss concept for the specific purpose of their work and in line with their particular industry, but they do not provide a definition for near-miss analysis. Near-miss analysis often refers to the process of identifying near misses and determining their root cause with a view to preventing and predicting accidents (Phimister et al. 2004).

Indeed, various accident investigations have revealed that almost all major accidents were preceded by a number of minor accidents and an even higher number of near misses as precursors (Mürmann and Oktem 2002). This is shown in Fig. 3.1 in the popular safety pyramid (Bird and Germain 1996). Recognising and handling these signals before an accident occurs have the potential to prevent an accident sequence from unfolding (Saleh et al. 2013) and to improve safety by providing valuable information about potential accidents (Phimister et al. 2004).

A working definition of near-miss analysis with regard to software systems follows:

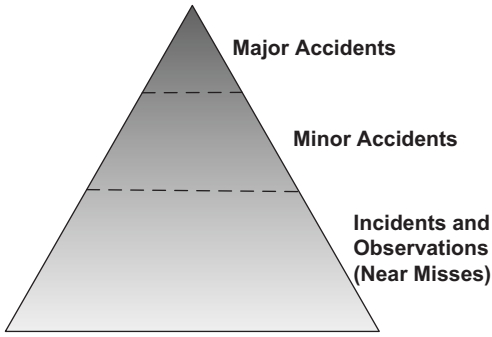

Fig. 3.1 The safety pyramid (Adapted from Bird and Germain 1996)

the root-cause analysis of near misses to prevent major software failures and understand their underlying causes.

Near-miss analysis is based on the observation that near misses and accidents have common causes but different outcomes (Andriulo and Gnoni 2014). This is due to the fact that a near miss is an immediate precursor to the impending accident. It is literally one step away from an accident. Therefore, an accident and a related near miss have the same sequence of leading events – the only difference is that in the case of a near miss, the sequence was interrupted just before the accident occurred.

Due to the interruption in the accident sequence, near misses either result in no loss or the loss incurred is minimal, contrary to what happens in the case of accidents. Identifying the cause of a near miss is therefore a valid method of identifying the cause of the ensuing accident. It has the additional benefit that learning about the accident is conducted without first incurring the loss caused by an accident. Besides, if properly recorded, the data pertaining to the accident sequence is available and complete since it has not yet been affected by the potential accident.

3.3 Tools and Techniques Used in Near-Miss Analysis

As a safety enhancement tool, near-miss analysis is often a component of a near-miss management program that is integrated with other safety management systems in an organisation. Besides the identification and analysis of near misses, a near-miss management program also includes activities for disseminating information about near misses to decision makers and for implementing countermeasures (Phimister et al. 2000). This process varies from one industry or organisation to the next. It is often done manually but can be automated through an electronic system commonly referred to as a near-miss management system (NMS). Near-miss management system is an umbrella term used to refer to software systems used to record, analyse and track near misses (Oktem 2002). They are sometimes referred to as near-miss systems. An effective NMS aims to quickly recognise near misses from the business operations in order to apply prevention measures. In order to be

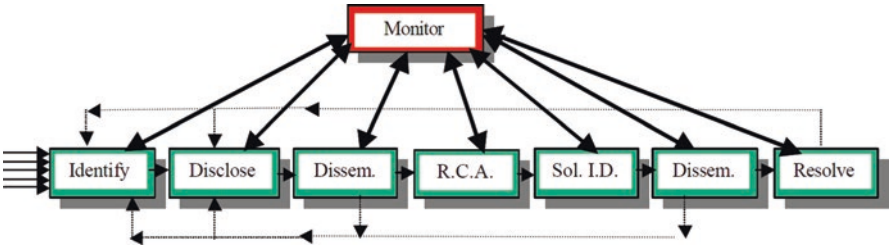

Fig. 3.2 Near-miss management process (Phimister et al. 2000)

effective, an ideal NMS is required to perform all activities of a near-miss management program. These activities are summarised in the following seven phases (Mürmann and Oktem 2002; Phimister et al. 2004):

1. Identification (recognition) of a near miss
2. Disclosure (reporting) of the identified near miss to the relevant people
3. Distribution of the information to decision makers
4. Root-cause analysis (RCA) of the near miss
5. Solution identification (remedial actions)
6. Dissemination of actions to the implementers
7. Resolution of all open actions and completion of reports

The above seven-stage process is illustrated in Fig. 3.2.

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 (Phimister et al. 2000). A review of the literature on the design of industry-specific NMSs indicates that most NMSs are single. Significant research has been conducted on the design of effective single NMSs (Wu et al. 2010; Andriulo and Gnoni 2014; Goode et al. 2014), especially in the healthcare industry for improved patient safety (Barach and Small 2000; Callum et al. 2001; Aspden et al. 2004; Fried 2009). Single NMSs usually place a strong emphasis on the identification and disclosure (reporting) phases of the near-miss management process described above. As such, they are often called *near-miss reporting systems* and are sometimes limited to that functionality of an NMS (Murff et al. 2005; Goode et al. 2014). Barach and Small (2000) provide a comprehensive list of proprietary near-miss reporting systems in various industries, which provide a user interface where users can enter various details about an observed near miss. Near-miss reporting systems are sometimes called *incident reporting systems* (Macrae 2007).

Apart from proprietary "private" near-miss reporting systems, some commercial NMSs are publicly available on the market. Commercial NMSs are mostly industry specific. Examples include AlmostME, a near-miss reporting system (Napochi 2013) for the medical field, and dynamic risk predictor suite (Near-miss Management LLC 2014), a comprehensive NMS designed for manufacturing facilities.

Near-miss analysis essentially consists of two steps: identifying near misses and determining their root cause. Techniques used for the identification of near misses

are often done manually by means of observation. Recognising an observed event or condition as a near miss requires a clear definition of 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 and that is easily understandable by all employees (Ritwik 2002; Phimister et al. 2003). Besides the manual identification, some effort has also been made to detect near misses automatically through an NMS, although limited literature is available on this subject. The strategy for the automatic detection of near misses is organisation specific but usually involves the specification of thresholds for various parameters that describe risky conditions (Wu et al. 2010). This requires historical data about past accidents and near misses.

Techniques for determining the root cause of a near miss include causal analysis and can be performed with investigation techniques taken from engineering disciplines, such as fishbone diagrams, event and causal factor diagrams, event tree analysis, fault tree analysis, failure mode and effects analysis (Phimister et al. 2003; Jucan 2005; Hecht 2007; RealityCharting 2013). The investigation consists of answering a series of questions that give insight into the factors that led to the near miss, the possible adverse consequences of the near-miss and the factors that prevented or limited those consequences. The investigation can be assisted by various tools such as a comparative timeline to organise data and various matrices such as the missed opportunity matrix and the barrier analysis matrix (Corcoran 2004).

Statistical analysis has also been proposed for learning from near misses. Some examples are using estimation techniques, simulations and regression analysis (Mürmann and Oktem 2002). Historical near-miss data can be used to estimate the loss distribution, i.e. the likelihood of a failure and its losses within a specific timeframe. Regression analysis can help determine exacerbating factors such as the frequency of certain operations. This information can then be used for simulating possible accident scenarios (Mürmann and Oktem 2002).

As valuable as the above root-cause analysis techniques are, they do not adhere to forensic principles. Thus they are not suitable for the forensic investigation of software failures. To this end, it is suggested that the methods and techniques of digital forensics be used as specified in the failure investigation process to analyse near misses – the same way as software failures will be analysed.

Although new to digital forensics, the value of analysing near misses has been recognised in various engineering disciplines, and near-miss analysis has been conducted for over three decades.

3.4 Benefits of Near Miss-Analysis Over Failure Analysis

The two main reasons for suggesting the use of near-miss analysis to complement the failure investigation process are discussed below:

- Severe system damage can impede accurate event reconstruction. In particular, volatile data that could pinpoint the root cause of the problem may be destroyed or corrupted after a system crash. Near-miss analysis provides an opportunity to proactively collect evidence, including volatile data, of the failure before it actually unfolds. This limits the risk of having evidence destroyed due to the failure.
- Severe failures from which valuable lessons can be learnt do not occur frequently. The opportunity to learn from them is therefore limited. Besides, many failures are not reported. This underreporting further reduces the pool of serious failures to learn from. In contrast, near misses can be numerous, thus they offer ample opportunity to learn more from their richer data sets. More cases of near misses also provide more evidence of a particular weakness in a system.

Moreover, it is generally agreed that in many cases accident precursors can be analysed more effectively than accidents – for the following reasons (Oktem et al. 2010; Ritwik 2002):

- Investigation of a severe failure is costly and time consuming. Hence, financial constraints and resource limitations can severely limit the depth of the investigation. Near misses are also smaller in size and easier to deal with than serious accidents.
- Legal concerns may affect the investigation adversely. For instance, in product liability litigations, organisations may well withhold information that could penalise them.

Reporting near misses has also been a legislative recommendation in the European Union since 1997 under the Seveso II Directive (Seveso II 1997). These events are to be reported in MARS (major accident reporting system), the mandatory reporting system for major industrial accidents within the European Union (Jones et al. 1999). In its Annex VI, the Seveso II Directive (Seveso II 1997: 33) makes the following recommendation:

> Accidents or "near misses" which Member States regard as being of particular technical interest for preventing major accidents and limiting their consequences (…) should be notified to the Commission.

3.5 Benefits of Analysing Near Misses Instead of Earlier Precursors

Investigating near misses is more valuable than investigating other early accident precursors for the following reasons:

- Various studies show that the number of precursors to an accident can be considerable (Borg 2002; Bird and Germain 1996). Selecting only the near misses reduces the number of precursors to be investigated, which can save resources required for the investigation.

- As a near miss is closer to the complete accident sequence, investigating a near miss provides the most complete preemptive evidence about the associated accident. It can therefore be used to identify the most accurate root cause of that accident.
- Early precursors can result in a number of false alarms, as they are further away from the unfolding of the accident. As they are closer to the accident, near misses provide the highest level of confidence about the imminence of an accident, which can lead to the implementation of the most relevant countermeasures.
- Early precursors are generally events and conditions that have been observed in the past and are easily identifiable. Near misses, on the other hand, are not predefined as they can vary from one accident scenario to the other. They are therefore best suited to identify new failure modes and possibly prevent them.

3.6 Near-Miss Analysis Success Stories

Near-miss analysis is used to help improve the reliability of a system, product or process by reducing its risk exposure to a potential disaster. It has a successful track record in organisations where it has been effectively implemented. For instance, evidence shows that near-miss analysis contributed significantly to the improvement of safety in the aviation industry (Phimister et al. 2004). Other examples with measurable benefits are discussed below:

- Studies from Norsk Hydro, a Norwegian aluminium company, show that when near-miss analysis was introduced in the organisation in 1985, the number of near misses reported went from 0 to 1800 within 13 years. This resulted in a reduction of lost-time injuries by around 75% (Jones et al. 1999).
- In Canada, a petroleum company reduced injury by 80% over a year and by 100% over 4 years after implementing a near-miss reporting program in 1986 (Borg 2002).
- In Malaysia, an oil company experienced a reduction in the monthly average cost of equipment-related accidents from $675,000 to $80,000 within a year after a near-miss reporting program was introduced in 1994 (Borg 2002).

3.7 Challenges to Near-Miss Analysis in the Software Industry

As beneficial as it is, near-miss analysis also has challenges that need to be addressed before its expected benefits can be reaped in the software industry. Across industries, the successful application of near-miss analysis faces two main challenges: (1) the detection of events and conditions that can be classified as near misses and (2) the high volume of near misses.

The detection of events as near misses is compounded by the difficulty in understanding what exactly constitutes a near-miss event. Furthermore, this varies between industries and organisations, which implies that a near miss can easily remain invisible to the untrained eye or system. For this reason, organisations need to define clearly and specify exactly which type of hazardous events should be reported as near misses. In many industries, near misses are obtained from observed physical events and conditions. However, in the case of software applications, some near misses might not be visible as no system failure occurs, which makes the detection task particularly challenging in the software industry. With regard to the high volume of near misses, it is generally understood that they can be frequent. In actual fact, they can be as much as 7–100 times more frequent than accidents (Aspden et al. 2004). In the hydrocarbon process industry, the accepted ratio of severe injury to near miss is between 15 and 25 (Ritwik 2002). More impressively, an extensive study of industrial accidents conducted in 1969 indicates that a severe injury can have up to 600 near misses as precursors (Nichol 2012). This is shown in the popular accident ratio triangle in Fig. 3.3. Another study suggests that this number could be even higher (Nichol 2012). A high volume of near misses is also expected in the software industry, as shown by reports of various major software failures.

The rich data set of near misses presents both a good learning opportunity and a challenge for investigation. With limited investigative resources, the sheer volume of near misses can become unmanageable. An effective data reduction mechanism is therefore required to investigate only those events as near misses that offer the most valuable learning opportunity and the most relevant evidence of the failure. This, among other things, requires an effective classification and prioritisation scheme.

3.8 A Structured Approach Towards Applying Near-Miss Analysis for Software Failure Investigations

Much work has been performed from both industry and academia to define a structured approach for near-miss analysis. One of the main components of such an approach should focus on the detection process usually based on assessing the risk level of an observed unsafe event or calculating the likelihood that this event can lead to a failure. Therefore, a thorough understanding of quantitative as well as qualitative methods is important for developing a successful structured approach for applying near-miss analysis for software failure investigations.

Much of the industrial work on near-miss detection is quantitative in nature and originates in the nuclear and aerospace industries with the American National Aeronautics and Space Administration (NASA) (NASA 2006) and the Nuclear Regulatory Commission (NRC) (Phimister et al. 2004). The NRC in particular published a number of study reports on using Bayesian statistics to determine the risk of a severe accident, based on operational data of observed unsafe events (Belles

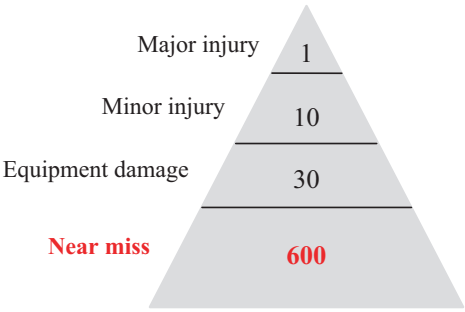

Fig. 3.3 Bird's accident ratio triangle (Adapted from Nichol 2012)

et al. 2000). Examples of such events include the degradation of plant conditions and failures of safety equipment (Belles et al. 2000). Bayesian statistics are used to calculate the conditional probability of an accident given the occurrence of the risky event. Significant research has also been conducted in other industries to find generic metrics or signs of an upcoming accident, such as equipment failure rates, failures of some system components or failures of safety systems (Leveson 2015). Probabilistic risk analysis (PRA) is also a recurring suggestion. PRA consists of estimating the risk of failure of a complex system by breaking it down into its various components and determining potential failure sequences (Phimister et al. 2004). In all the above work, near misses are usually identified as those events that exceed a predefined level of severity.

With regard to qualitative analysis for near-miss detection, most of the existing approaches are simply based on a listing of potential hazards such as a toxic chemical leak or an improperly closed switch box (Ritwik 2002). Some organisations use information obtained from incident reporting systems to identify potential near misses. A root-cause analysis of the incidents is conducted to identify the factors that led to the event. Another technique also used is the Delphi method. The Delphi method is a group decision-making tool that can be used to obtain information on the probability of an accident from a panel of experts (Phimister et al. 2004).

Although they all have some merit, both the quantitative and qualitative approaches have disadvantages that limit their application to the investigation of software failures. For instance, the validity of the quantitative analysis techniques depends heavily on the risk threshold set for near misses. A high threshold may overlook significant events that were not anticipated, especially in new or immature software systems, while a low threshold will likely result in many false alarms (Phimister et al. 2004). Besides, generic metrics of near misses might not be applicable to all types of systems and all types of failures. In addition, quantitative analysis techniques often assume that accidents are caused by a linear chain of events or by component failures, which is not always the case for software failures. In qualitative analysis, a listing of precise near-miss conditions can leave out a number of risky conditions that have not been experienced before. This book provides a more flexible approach to detect near misses for software failures. This near-miss approach relies on the identification of near-miss indicators. Near-miss indicators are expected to be identified from an analysis of the changes in the system's behav-

iour converging towards the failure. Indeed, it is generally accepted that accidents occur after some kind of behaviour or change (Leveson 1995). For example, a sharp increase in the number of dropped packets can indicate an impending failure due to network congestion and can be considered a near-miss indicator in a specific network set-up. An exception to the above assumption about near-miss indicators is the case of a failure that occurs abruptly with no warning signs (e.g. due to a power failure). The identification of near-miss indicators relies on a comparison of the patterns between the expected system behaviour and the system's behaviour just before the failure.

The initial pattern analysis can only be performed after a failure has occurred, and it is specific to that failure and to the system at hand. However, once a formula has been created from the identified near-miss indicators, the formula can be used in the subsequent monitoring of the system and can automatically detect near misses, of that type, before the recurrence of a failure. The formula is adjusted and fine-tuned over time as near misses are detected. A high-level description of this approach, illustrated in Fig. 3.4, follows:

- Depict and analyse patterns in the failure logs from end-to-end (from when the system starts running to when it fails).
  - If the patterns are the same throughout, stop as no near-miss indicator has emerged.
  - If behavioural patterns change throughout the execution of the program, identify point of change
- Profile behaviour from point of change to failure.
- Compare to expected behaviour after failure.
- Use the identified change in behaviour close to the point of failure (point 4) to define near-miss indicators.
- Establish interdependencies between near-miss indicators.
- Use the interdependencies between the various indicators to create a near-miss formula.
- Apply the formula to a running system to detect subsequent near misses.

It is worth noting that the approach depicted in Fig. 3.4, for the first time when it is executed, is not a fully automated process. In particular, the creation of a near-miss formula is a semiautomated task using the rules generated by machine learning (pattern analysis) algorithms. Once the near-miss formula is known, it can then be used to automatically detect near misses in the future. The near-miss formula is designed to be specific for a type of software failure. Furthermore, it can be said that in general, industry-wide near-miss indicators are not effective, and therefore system-specific indicators are preferable. The proposed approach (Fig. 3.4) can be seen as an adaptation of the accident investigation technique called change analysis. Change analysis is used to analyse the difference between what was expected and what actually occurred. It compares an accident scenario to the accident-free situation used as a baseline (Sklet 2002). The identified differences are analysed to determine how they contributed to the accident. Contrary change analysis does not

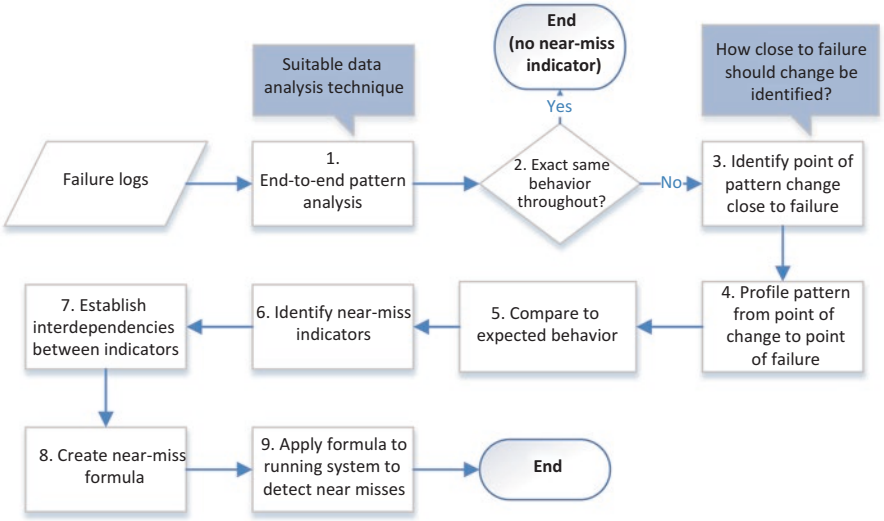

Fig. 3.4 High-level flowchart of proposed near-miss approach

specifically look for changes close to the time window during which the accident appears. The approach presented in Fig. 3.4 also differs from other near-miss detection techniques that usually make use of failure probabilities and threshold calculations. When compared to these techniques, the present approach is simpler and closer to reality as it presents the actual events that occurred instead of hypotheses about potential events or consequences. However, this simplicity does not make it less effective.

Implementing the proposed near-miss approach gives rise to the following questions: which data analysis techniques to use for the pattern analysis? How close to the point of failure are near-miss indicators identified? How to identify interdependencies between near-miss indicators? These questions are indicated in the coloured blocks (call-outs) in Fig. 3.4. The near-miss approach presented above relies on the analysis of failure logs, which can have a high volume. Since the goal of this analysis is to observe a pattern in the system's behaviour, a tool with powerful visualisation capability that can handle large data sets efficiently is required. For this reason, a tool delivering results in a visualised way is ideal.

The answer to how close to the point of failure are near-miss indicators are identified depends on the outcome of the pattern analysis of the failure logs. As soon as a significant deviation from operational expectation is observed in the last few entries in the log file before the failure, then this point in time can be considered as appropriate to start inspecting the logs for near-miss indicators. Identifying the exact point of change from expected to unexpected behaviour is likely to require some iteration of the process of comparing the expected behaviour to the behaviour close to the failure. The identification of interdependencies and relationships between near-miss indicators are based on iterative experimentation. This is

done by changing the set of variables during the iterations which is then analysed by investigating the behavioural patterns emanating from the failure logs. More details about this approach are provided in the description of the experiment in a later chapter.

3.9 Conclusion

This chapter presented near-miss analysis as a promising technique for dealing with the volatility of the digital evidence required to conduct a forensic investigation into software failures. As near-miss analysis is not yet used in digital forensics, the chapter emphasised its application and benefits in other industries to motivate its application in digital forensics. Challenges to near-miss analysis for such a purpose (i.e. the high volume of near misses and their difficult detection due to the fact that they are not easily observable in a software system) were also presented. A structured approach towards applying near-miss analysis for software investigations is presented.