Near Miss Detection for Software Failure Prevention

Madeleine A. Bihina Bella[1], Martin S. Olivier[2] and Jan H. P. Eloff[3]
ICSA Research Lab
Department of Computer Science
University of Pretoria[1,2,3], P. O. Box 0002, Pretoria, South Africa
Tel: +27 12 420 3653, Fax: +27 12 362 5188
and SAP Research Pretoria[1,3]
email: mbihina@yahoo.fr[1]; martin@mo.co.za[2]; jan.eloff@sap.com[3]

Abstract-The increased complexity of IT systems can lead to software operational failures with disastrous consequences. In order to correct and prevent the recurrence of such failures, a thorough post-mortem investigation is required to localize their root causes. However, this cannot be effectively addressed by existing failure investigation disciplines as they are reactive by definition and only applied after major accidents occur. We therefore suggest a proactive approach through the detection and analysis of near misses. Contrary to failures, near misses do not result in loss or damage but rather indicate a system weakness that could lead to a malfunction under different circumstances. Thus, they are warning signs that provide an opportunity to prevent serious failures and improve the system's reliability. Though near miss management is successfully implemented in other industries it is not yet used in the IT field. As near misses cannot be successfully managed without a proper detection and prioritisation mechanism, this paper proposes an IT specific definition of a near miss and then presents a high-level process for detecting and prioritising such events. We illustrate the application of the definition and prioritisation scheme with a practical example of a real-life mobile application developed by SAP Research Pretoria. We further outline some of the difficulties in the successful implementation of the process.

Index Terms— Near miss, failure analysis, root cause analysis, accident, incident

I. INTRODUCTION

Major software failures of IT systems often result in disasters including high financial loss, lengthy interruption of productivity, damage to public image, and litigation issues from disgruntled customers. In more extreme cases, it can even result in injury and loss of lives as was the case with the infamous Therac-25, a computer-controlled radiation therapy machine that seriously burnt and was partly to blame for the death of several cancer patients due to various software bugs between 1985 and 1987 [1]. More recently, the Blackberry outage in October 2011 cut off millions of BlackBerry smartphone users worldwide for 3 days due to a faulty backup system [2].

In order to prevent the recurrence of such severe accidents, a thorough post-mortem investigation must be conducted to identify their root causes [3]. This view is supported by various standards and best practices such as the ISO/IEC 27002 information security standard [4] and the American National Institute of Standards and Technology [5]. These standards further recommend the collection of forensic evidence to analyse the failures in the event they lead to litigations (e.g. product liability lawsuit). Two main disciplines have emerged in the last decade specifically for this purpose. These are forensic software engineering [6] and operational forensics [7].

Unlike failure analysis and root cause analysis of software products which tend to focus on analysing defects observed during system development, testing and deployment [8] [9], forensic software engineering and operational forensics analyse failure data from operational systems. However, while the former searches the "software engineering" causes of the failure – i.e. the factors in the software development process that led to or contributed to the malfunction [10] – the later focuses on the operational causes of the failure [11].

While each discipline has its merits, they are both reactive by nature and thus cannot prevent an accident from happening. Preventing major failures requires a proactive approach not addressed by existing disciplines in the IT industry. A promising solution, widely used in many other industries, is the analysis of near misses [12].

Unlike accidents, near misses do not result in loss or injury. They are instead unsafe situations with potential for loss or damage [13]. A simple example is a driver crossing a red light at a busy intersection without a collision. In the case of engineering products, near misses can indicate a system weakness that could lead to a costly failure under different circumstances [14]. A practical example is the case we experienced with the BiYP (Business in Your Pocket) mobile application developed by SAP Research Pretoria as part of the centre's mission to address issues of small enterprises in emerging economies through mobile technology [15]. BiYP enables small shops to order goods from multiple suppliers through their online and real-time product catalogues, which are available on the user (shop owner) mobile phone. Due to memory limitation on the phone, instead of the entire supplier's catalogue, each user receives a personalized version of a supplier's catalogue based on his buying patterns but can add new items as needed. As one user kept on adding new items to his catalogue, he exceeded the specified product number threshold, but this did not cause the application to crash, contrary to what was normally expected. Investigations

revealed that this was due to ineffective memory management in the application.

Various accident investigations have revealed that almost all major accidents had a number of prior minor incidents and an even higher number of near misses [16]. This is shown in Figure 1 in the popular safety pyramid, initially developed by Frank E. Bird, Jr. [17].

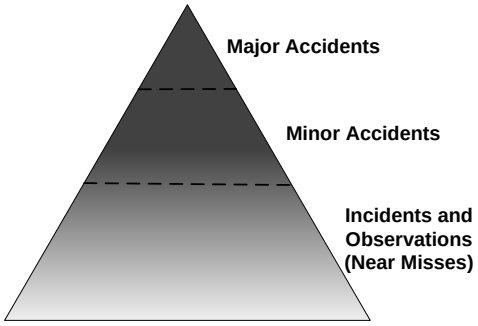

Figure 1: The Safety Pyramid [17]

Thus, near miss analysis can help improve the reliability of a system by reducing its risk exposure to a potential disaster [13].

Although not yet in use in the IT industry, near miss analysis has proved beneficial in various industries including the chemical, process, aviation, nuclear, military and healthcare industries [18]. Some interest has also been shown in its application in the financial industry [16] [13]. Near miss investigation is often performed through so called near miss management systems which are software systems used to record, analyse and track near misses [16]. The successful implementation of a near miss management system requires a clear definition of near misses to recognise them and effective methods to detect and prioritise them.

This paper proposes a definition of near miss suitable for IT systems that can help not only identify near misses but also rank them according to their risk level. The proposed definition is based on the concept of service level agreement (SLA) which defines the contractually agreed level of functionality of a system. In addition, we outline a high level process for detecting and prioritising near misses. We also discuss some difficulties in implementing the process and future research avenues to address them.

The remainder of the paper is organised as follows: Section 2 presents near miss opportunities, benefits and challenges. Section 3 reviews previous work on near miss analysis. Section 4 defines an IT specific definition of a near miss based on the concept of SLA and its relationship to failure. Our proposed near miss detection and classification process is discussed in Section 5. The paper ends with a conclusion and future work in Section 6.

II. Opportunities, Benefits and Challenges of Near Miss Management

It is generally agreed that precursor events can often be more effectively analysed than accidents for the following reasons [12]:

- Severe system damage can impede accurate event reconstruction. In software systems in particular, volatile data that could pinpoint the root cause of the problem may be lost after a crash.
- Investigation of a severe failure is costly and time consuming. Financial and resource limitations can thus severely limit the depth of the investigation.
- Legal concerns may adversely affect the investigation. For instance, in product liability litigation, organisations may withhold information that could penalise them.

Near miss analysis offers the following benefits over failure analysis [14] [13]: (1) a near miss is a cheaper learning tool than an actual failure, which can be very costly, as indicated earlier; (2) near-misses are smaller in size and easier to deal with than serious accidents; (3) severe accidents are generally sporadic and the opportunity to learn from them is rare. In contrast, near-misses can be numerous, thus we can learn more from their richer data sets. Their analysis can increase the certainty about the probability of an accident and lead to better corrective actions.

Furthermore, near miss analysis offers the following learning opportunities [12]. Firstly, it can reveal what can go wrong and how accidents can develop through modelling and simulations. This may help uncover new failure modes. Secondly, it can provide trends in the system reliability through monitoring. It can show whether reliability is improving, thus whether accident probability is decreasing. Finally, it increases awareness of system flaws or weaknesses.

Near miss analysis has a successful track record in organisations where it has been effectively applied. For instance, evidence shows that it contributed to the improvement of safety in the aviation industry [12]. Studies from Norsk Hydro, an aluminium company, show that when near miss management was introduced in the organisation, the number of near misses reported went from 0 to 1800 over 13 years (which corresponds to approximately 0.5 reports per employee per year) which resulted in lost-time injuries being reduced by around 75% [19].

However, near miss analysis also has its challenges. As they are numerous, they cannot all be investigated due to limited resources. They thus require an effective prioritisation and classification scheme. Another major challenge with the analysis of near-miss events is their detection. As they do not result in failures and no loss is incurred, they can remain invisible to the untrained eye or system. Finding solutions to these challenges has been the focus of previous work in the field of near miss management. This is reviewed in the next section.

III. Review of Previous Work on Near Miss Management

Near miss analysis has been in use for several decades in a number of industries [12]. When investigating near misses two fundamental questions arise: (1) how to select near

misses for investigation and (2) how to perform root cause analysis of the selected events [20]. Therefore most of the literature on this field revolves around these two aspects.

With regard to question (1), various approaches are used to classify and prioritise near misses. Besides manual screening, risk-based classification and mathematical modelling can be used. Risk-based classification ranks near misses based on the severity level of their potential consequences [14] or on their frequency [21]. This can be performed with a risk decision matrix [14] or historical data of reported near misses to determine trends in the occurrence of certain near miss events [12]. Mathematical modelling can be used to estimate the conditional probability of an accident given a near miss in order to assess its level of severity. Probabilistic risk analysis (PRA) and Delphi techniques can also be used for the same purpose [12]. PRA consists of estimating the risk failure of a complex system by breaking it down into its various components and determining potential failure sequences [22]. 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 [23].

With regard to question (2), like failure and root cause analysis, causal analysis of near misses can be performed with investigation techniques from engineering disciplines such as fishbone diagrams, event and causal factor diagrams, event tree analysis, fault tree analysis and failure mode and effects analysis [18] [24] [25]. The investigation consists of answering a series of questions that give insight into the factors that led to the near-miss, the near miss possible adverse consequences 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 [20]. Statistical analysis has also been proposed for learning from near misses. Some examples are using estimation techniques, simulations and regression analysis [16]. 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 [15].

In addition, two major research projects on near miss management provide a significant number of papers on the topic. The first one is the Near Miss Project at the Risk Management and Decision Processes Center at the Wharton School, University of Pennsylvania, which has been ongoing since 2000 [26]. 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 which extensively reviewed near miss management across industries to promote cross-industry knowledge sharing is available in an online book [12]. Several industry-specific programs were also established years before that. In addition, research has been conducted in this field several decades earlier. Some of this research has been published in workshop proceedings by [27] and [28].

This literature review shows that near miss management is not a new concept and a vast body of knowledge exists on this field, which has proved valuable. However, it has not yet received much attention in the IT industry. A proper definition for an IT system near miss is thus still lacking. We propose such a definition in the next section.

IV. NEAR MISS DEFINITION FOR SOFTWARE SYSTEMS

The exact definition of what constitutes a near miss varies from one industry and organisation to another. In order to detect and analyse near misses, we therefore need to give them a definition that is suitable for the IT industry and will help us not only recognise but also prioritise near misses.

As mentioned earlier, a near miss is a sequence of unsafe events that can lead to a failure. An example would be the loss of a spare server that would result in a crash in case its processing power is required by the workload at the time it is down. If the redundant server is recovered before the workload can no longer be handled by the remaining active servers, then this event is a near miss, otherwise it becomes a failure. The impact of the failure would determine whether this is a minor incident or an accident. This is clearly visible in Figure 2, in which we show how a near miss relates to the associated failure, incident and accident in terms of the time of the event and the loss incurred. Our argumentation is that a failure may or may not result in loss and becomes an accident only when significant loss is incurred.

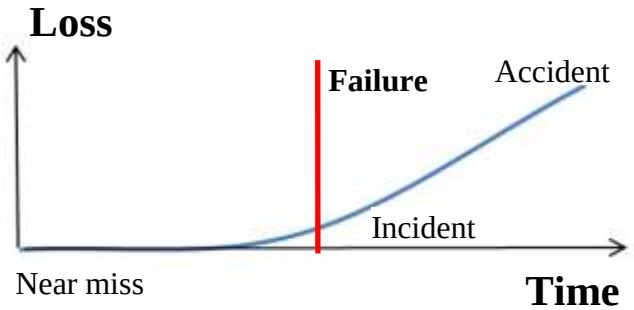

Figure 2: Relation between near-miss, failure, incident, and accident in terms of time of event and loss incurred

In a typical organisation, the above event would only be investigated if it resulted in a major accident with significant impact. We argue that since this could reoccur with high impact, it is worth investigating the reason for the servers' failures even though no loss was incurred. As evidence, referring back to the earlier example of the BiYP mobile application, although the reported near miss (exceeding the threshold for the number of products per user's catalogue) did not result in any loss for that specific user, it did reoccur with another user, whose application ended up crashing due to the memory overload. Interestingly, at the moment of the

crash, this user (user B) had fewer items in his catalogue than the first user (user A) whose application did not crash. Investigations showed that this was due to the fact that his catalogue items had bigger corresponding pictures than user A's catalogue. Therefore, his catalogue memory usage was higher than user A's catalogue. This indicated that the varying picture size of items needed to be taken into account when specifying the threshold for the number of products per catalogue. It is possible that this problem could have affected more users if the cause of the near miss (and subsequent crash) had not been investigated and corrected.

As can be seen from the above examples, the near miss events would not be visible to the end user as no failure would be observed. This could make their detection challenging and only possible through system monitoring. Since near misses are closely related to failures, the definition of a failure is used as the basis.

A failure is the inability of a system or component to perform its required functions within specified performance requirements [29]. These requirements are usually specified in a service level agreement (SLA) between the service provider and the customer. The SLA is the entire contract that specifies what service the customer can expect from the provider, and the responsibilities of both parties [30]. The SLA comprises a number of Service Level Objectives (SLOs), which are specific performance metrics for the service such as availability, throughput, frequency and response time. SLOs are usually expressed in terms of an achievement value, a target measurement and a measurement period. Where and how they are measured is also sometimes part of the specification [29]. For instance, the SLO for the availability of the BiYP application may specify that the mobile catalogue will be operational and available to the customer at least 99.9% of the time in any calendar month.

If the performance of the system no longer meets a specified SLO, it is no longer useful to its users, and the system is considered down for practical purposes [31]. Using the above example, the SLO indicates that the system should not be down for more than 0.1% of the time in a month. For a 30 day month, this corresponds to 0.03 day or 43min and 12s. If the total period of time when the mobile catalogue was unavailable exceeds this limit in a given month, then the SLO has been violated. The system is considered to have underperformed and thus to have failed. Therefore, a failure can be defined as the violation of a specified SLO. It results that *a near miss is an event or unsafe condition that has the potential to lead to an SLO violation.*

This definition is broad enough to encompass any event or situation that may lead to a failure i.e. a poorly performing system. Using an SLO as a measurable characteristic in the definition provides a way to quantify the severity of an unsafe condition and to prioritise near misses. As an illustration, still using our earlier example of the 99.9% availability SLO, if the system was down for close to 43min and 12s in total in a month, but not more than that, then this situation is a near miss. How close the downtime was to this figure will help to assign it a risk level.

Once near misses are clearly defined, they can be recognised and need to be classified to facilitate their analysis and to benefit from this analysis. The next section presents our process to perform these steps.

V. NEAR MISS DETECTION AND CLASSIFICATION PROCESS

Various papers prescribe the following 8 step process for successfully managing near misses [12] [26] [16]:

- 1: Identification and recognition of a near-miss
- 2: Disclosure (reporting) of the identified information/incident
- 3: Prioritization and classification of information for future actions
- 4: Distribution of the information to proper channels
- 5: Analyzing causes of the problem
- 6: Identifying solutions (remedial actions)
- 7: Dissemination of actions to the implementers and general information to a broader group for their knowledge
- 8: Resolution of all open actions and completion of reports

These steps can be automated through a near miss management system or they can be performed manually. This paper focuses on Steps 1 and 3, which correspond to the current stage of our research. Our suggestions to perform these steps are discussed next.

A. Step 1: Identification and recognition of a near-miss

Our partial solution for Step 1 was presented previously in Section IV with the definition of a near miss. Clear examples of possible near miss scenarios can further help recognise them. Detection is enabled through system monitoring. Our suggestion is to monitor the system and record event logs in a central repository such as a Syslog server. By default, Syslog messages are assigned a facility code and a severity value [32]. The facility code indicates the source of the message (e.g. printer, network) and the severity value indicates its criticality (e.g. warning, error, emergency). This information can be used for a primary categorisation and ranking of unsafe situations. Only highly severe events from critical resources will be labelled as potential near misses.

B. Step 3: Prioritization and classification of information for future actions

Regarding Step 3, we need to quantify the severity of a near miss so that only near misses with a high risk level are passed on for in-depth investigation. We suggest specifying a near miss threshold that indicates how close the near miss should be from violating the SLO to be considered high risk. This threshold will vary from one organisation to another depending on their risk tolerance. For instance, user A might be comfortable with a 95% threshold while user B will limit its tolerance level to 75%. Using the BiYP example in

Section 4, this would correspond to a total monthly downtime of 41min and 2s (95% of the limit) for user A and to 32min and 24s (75% of the limit) for user B. This prioritisation scheme can be formally expressed with the following mathematical notation.

$D_{experienced}$ is the experienced downtime
$D_{allowed}$ is the maximum permissible downtime according to the SLO (43min and 12s in our case)
α is the near-miss threshold
If $D_{experienced} \geq \alpha \times D_{allowed} \rightarrow$ high risk near-miss

In addition to the near miss threshold, we are considering using mathematical modelling to calculate the conditional probability of a failure given an unsafe event occurring. This would provide a triage mechanism of near misses for further investigation. This method of classification can be expressed as follows:

$D_{expected}$ is the expected downtime i.e. the expected loss of productivity due to a failure
P is the probability of failure given the current unsafe situation
MTTR is the mean time to repair the system in case it goes down
$D_{expected} = P \times MTTR$
If $D_{expected} \geq \alpha \times D_{allowed} \rightarrow$ high risk near-miss that should be investigated

The MTTR can be obtained from historical observation or through the system vendor's specifications. The expected downtime is the product of the failure probability and the MTTR. We are currently exploring concepts and formulas from the reliability and availability theory of software systems [33] to develop our failure probability formula.
The integration of the above steps is illustrated in Figure 3.

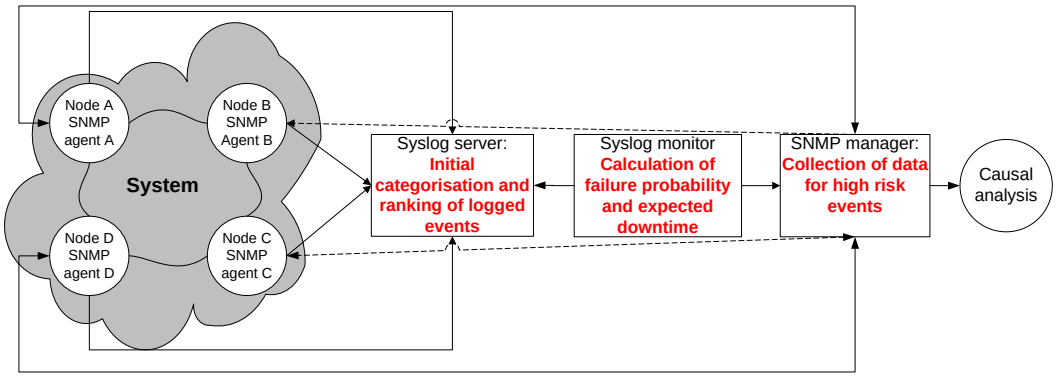

Figure 3: The near miss detection and classification process

Figure 3 is an illustration of the process described earlier for a system comprising 4 nodes. Solid lines indicate mandatory communication between components while the dotted lines show possible communication links. The system is represented by the shaded cloud and nodes are numbered from A to D. Each node sends its event logs to a Syslog server. The Syslog server does a primary categorization and ranking of the logged events using their facility codes and severity values. Events flagged critical are then sent to another module or server (we named it Syslog monitor) which calculates the failure probability and expected downtime from each event. Events classified as high risk are passed on to another component such as an SNMP manager which requests and obtains additional information about the event and its source from the relevant nodes through their SNMP agents. SNMP is an Internet-standard protocol that enables this information exchange [34]. This data is then used for causal analysis of the near misses.

A benefit of this approach is that it enables the safe and proactive collection of data related to the potential failure, which can be challenging after a failure due to volatile data being erased or corrupted. It also enables the prevention of a serious failure once the cause of the unsafe event has been established and corrective actions have been taken.

The next logical step would be to apply the BiYP application to this process. However, space limitations prevent this in this paper. It will thus be left for future work.

I. CONCLUSION

The paper presents the concept of near miss analysis, its application in other industries and how it can be used to prevent major software failures in the IT industry. We have proposed a definition of near misses suitable for IT systems and suggested a process to detect, classify and prioritise near miss events. We illustrated its application with the real life example of a mobile application to order products online through a mobile phone. This process is work in progress and research is underway to refine it. Future work is the development of a mathematical formula to calculate the likelihood of a failure due to a near miss in order to prioritise high risk near misses for detailed causal analysis.

II. ACKNOWLEDGEMENT

The support of SAP Research Pretoria/Meraka CSIR towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are solely those of the authors.

III. REFERENCES

[1] N. Leveson and C.Turner. "An Investigation of the Therac-25 Accidents". *IEEE Computer*, vol. 26, pp.18-41, Aug. 2002.

[2] A. Ahmed. "BlackBerry service down". Eyewitness News (Oct. 10, 2011). [Online]. Available: http://www.ewn.co.za/Story.aspx?Id=75494 [Oct. 15, 2011].

[3] B. Meyer. "Again: The One Sure way to Advance Software Engineering". *ACM communications blog*. Internet: http://cacm.acm.org/blogs/blog-cacm/101891-again-the-one-sure-way-to-advance-software-engineering/fulltext, (Jan. 13, 2011) [Feb. 17, 2012].

[4] ISO/IEC. "Information technology - Security techniques - Code of practice for information security management". Switzerland, Geneva. International Standard ISO/IEC 27002, 2007.

[5] K. Scarfone, T. Grance and K. Masone. "Computer Security Incident Handling Guide". Gaithersburg, USA. National Institute of Standards and Technology Special Publication 800-61, (Revision 1). [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf, Mar. 2008 [Feb. 22, 2011].

[6] L. Hatton. "Forensic Software Engineering: an overview". CIS, University of Kingston, UK. Internet: http://www.leshatton.org/wp-content/uploads/2012/01/fse_Dec2004.pdf, Dec. 19, 2004 [May 05, 2012].

[7] M. J. Corby. "Operational Forensics" in *Information Security Management Handbook*, 6th ed, H.F. Tipton and M. Krause, Auerbach Publications, Boca Raton, 2007, pp.2773-2779.

[8] V.B. Mendiratta, J.M. Souza and G.H. Zimmerman. "Using Real-World Software Failure Data", presented at the Workshop on Reliability Analysis of System Failure Data, Cambridge, UK, 2007.

[9] R.B.Grady. (1996, Aug). "Software Failure Analysis for High-Return Process Improvement Decisions". *Hewlett-Packard Journal*, [Online]. Available: http://www.hpl.hp.com/hpjournal/96aug/aug96a2.pdf, [May 10, 2012].

[10] C. Johnson. "Forensic software engineering: are software failures symptomatic of systemic problems?". *Safety Science*, vol. 40 (issue 9), pp. 835–847, Dec. 2002.

[11] M.A. Bihina Bella, M.S. Olivier and J.H.P. Eloff. "Proposing a Digital Operational Forensic Investigation Process", In Proc. *6th Workshop on Digital Forensics and Incident Analysis (WDFIA)*, 2011.

[12] J.R. Phimister, V. Bier and H. Kunreuther. *Accident Precursor Analysis And Management: Reducing Technological Risk Through Diligence*. Washington D.C, USA: The National Academies Press, 2004.

[13] U. G. Oktem, R. Wong and C. Oktem. "Near-Miss Management: Managing the Bottom of the Risk Pyramid". *Risk & Regulation*, magazine of the ESRC Centre for Analysis of Risk and Regulation, Special Issue on Close Calls, Near Misses and Early Warnings, pp.12-13, Jul. 2010.

[14] U. Ritwik. "Risk-based approach to near miss". *Hydrocarbon processing*, pp. 93-96, Oct. 2002.

[15] S. Cashmore. "Adding muscle to mobile apps". *Brainstorm IT Web*, vol.11, issue 08, pp. 38-39, Apr. 2012.

[16] A. Mürmann and U. Oktem. "The near-miss management of operational risk". *The Journal of Risk Finance*, vol. 4, issue 1, pp.25-36, Jul. 23, 2002.

[17] F.E. Bird Jr. and G.L. Germain. *Practical Loss Control Leadership*. Loganville, Georgia: Det Norske Veritas Inc. 1996.

[18] J.R. Phimister, U. Oktem, P.R. Kleindorfer and H. Kunreuther. "Near-miss Incident Management in the Chemical Process Industry". *Risk Analysis*, vol. 23, no. 3, pp.445-459, 2003.

[19] S. Jones, C. Kirchsteiger, and W. Bjerke. "The Importance of Near Miss Reporting to Further Improve Safety Performance". *Journal of Loss Prevention in the Process Industries*, vol. 12 pp. 59-67, 1999.

[20] W.R. Corcoran. "Defining and Analyzing Precursors", in *Accident Precursor Analysis And Management: Reducing Technological Risk Through Diligence*, J.R. Phimister, V. Bier and H. Kunreuther, Washington D.C, USA: The National Academies Press, 2004, pp. 79-84.

[21] W.S. Greenwell, J.C. Knight, and E.A. Strunk. "Risk-Based Classification of Incidents". In *Workshop on the Investigation and Reporting of Incidents and Accidents*, Department of Computer Science, University of Virginia, 2003. [Online]. Available: http://shemesh.larc.nasa.gov/iria03/p03-greenwell.pdf [May 7, 2012].

[22] W. E. Vesely. "Probabilistic Risk Assessment" in *System Health Management: With Aerospace Applications*, S. B. Johnson, T. J. Gormley, S. S. Kessler, C. D. Mott, A. Patterson-Hine, K. M. Reichard and P. A. Scandura, Chichester, UK: John Wiley & Sons, 2011.

[23] H.A. Linstone and M. Turoff . (2002). *The Delphi Method: Techniques and Applications*. (electronic version). [On-line]. Available: http://is.njit.edu/pubs/delphibook/delphibook.pdf [May 17, 2012].

[24] G. Jucan. "Root Cause Analysis for IT Incidents Investigation". Internet: http://hosteddocs.ittoolbox.com/GJ102105.pdf, 2005 [Oct. 10, 2010].

[25] M. Hecht. "Use of software failure data from large space systems", presented at the Workshop on Reliability Analysis of System Failure Data, Cambridge, UK, 2007.

[26] J.R. Phimister, U. Oktem, P.R. Kleindorfer and H. Kunreuther. "Near-Miss System Analysis: Phase I". Wharton School, Center for Risk Management and Decision Processes. Internet: http://opim.wharton.upenn.edu/risk/downloads/wp/nearmiss.pdf , 2000 [Jul. 19, 2011].

[27] T.W. Van der Schaaf, D.A. Lucas, and A.R. Hale. *Near-Miss Reporting as a Safety Tool*. London: Butterworth-Heinemann, 1991.

[28] V.M. Bier. *Accident Sequence Precursors and Probabilistic Risk Assessment*. Madison, Wis.: University of Wisconsin Press, 1998.

[29] IEEE. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. 1990.

[30] D.C. Verma. "Service Level Agreements on IP Networks", in Proc. IEEE, vol. 92, no. 9, 2004, pp.1382-1388.

[31] W.H. Highleyman. "Configuring to Meet a Performance SLA – Part 1: Single Server Response Times". Internet: http://www.availabilitydigest.com/, 2008 [Mar.1, 2012].

[32] R. Gerhards. "The Syslog Protocol", RFC 5424, Internet: http://tools.ietf.org/html/rfc5424, 2009 [Jul. 29, 2011].

[33] B. Holenstein, B. Highleyman and. P.J. Holenstein. *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, vol. 1, Bloomington, USA: Authorhouse, Dec. 2003, pp. 27-28.

[34] R. Presuhn. "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)". RFC 3418. Internet: http://tools.ietf.org/html/rfc3418, Dec 2002 [Jul. 29, 2011].

Madeleine Bihina Bella is pursuing her PhD in Computer Science at the University of Pretoria. She works as a Research Associate for SAP Research, is a Certified Information Systems Auditor and a 2011 L'Oreal/UNESCO for Women in Science regional fellow. Her research interests include computer fraud detection and digital forensics.