AMBER

Assessing, Measuring, and Benchmarking Resilience

FP7 - 216295

Project no.: 216295

Project full title: Assessing, Measuring, and Benchmarking Resilience

Project Acronym: AMBER

Deliverable no.: D2.2

Title of the deliverable: State of the Art

Preparation Date of Delivery: June 30th, 2009

Approval Date of Delivery: June 30th, 2009

Organisation name of lead contractor for this deliverable: University of Newcastle upon Tyne

Author(s): Aad van Moorsel, Eugenio Alberdi, Raul Barbosa, Robin Bloomfield, Andrea Bondavalli, João Durães, Rosaria Esposito, Lorenzo Falai, Johan Karlsson, Paolo Lollini, Henrique Madeira, Istvan Majzik, Zoltán Micskei, Leonardo Montecchi, Gergely Pinter, Vladimir Stankovic, Lorenzo Strigini, Michele Vadursi, Marco Vieira, Katinka Wolter, Huqiu Zhang

Participant(s): All partners

Work package contributing to the deliverable: WP2

Nature:

Version: 2.0

Total number of pages: 250

Contract Start Date: 1 January 2008

Duration: 24m

Project Coordinator: University of Coimbra (Portugal)

Partners: University of Coimbra, Budapest University of Technology and Economics, Chalmers University of Technology, City University London, University of Newcastle upon Tyne, Universitá degli Studi di Firenze, and Resiltech

Project funded by the European Community under the Seventh Framework Programme

State of the Art

Authored by: Aad van Moorsel, Eugenio Alberdi, Raul Barbosa, Robin Bloomfield, Andrea Bondavalli, João Durães, Rosaria Esposito, Lorenzo Falai, Johan Karlsson, Paolo Lollini, Henrique Madeira, Istvan Majzik, Zoltán Micskei, Leonardo Montecchi, Gergely Pinter, Vladimir Stankovic, Lorenzo Strigini, Michele Vadursi, Marco Vieira, Katinka Wolter, Huqiu Zhang

June 30th, 2009

Table of Versions

Version	Date	Authors	Version Description	Reviewers	Date of Approval
0.1	June 1, 2009	Aad van Moorsel	Deliverable 2.2		
1.0	June 27, 2008	Aad van Moorsel	Deliverable 2.2	Partners reviewed individual chapters	
2.0	June 30, 2009	Aad van Moorsel	Deliverable 2.2	Integrated reviewed deliverable	

Table of Contents

Abstract

This report provides a review of the state of the art in resilience assessment, measurement and benchmarking (AMB) technologies for computer and information systems. It serves as input to the research agenda in information and computer system resilience assessment, to be produced by the AMBER (Assessing, Measuring and Benchmarking Resilience) Coordination Action. This document is the final version of the state of the art reports, dealing with the following areas:

- relevant European projects, research agendas and standardisation bodies
- core AMB technologies, including fault injection, resilience benchmarking, modelling and model-based assessment, online monitoring, security assessment, safety and assurance cases, robustness testing and human factors
- technologies of general importance to the area, including meterology, field measurement, and analysis of trends in resilience assessment and data analysis and presentation

Each chapter contains a historical perspective on the discussed topic, highlighting the most important contributions in the field. At the end of the document, a list of about 700 references provides the main literature in the area. The Introduction provides a high level overview of the major achievement and breakthroughs in computer systems AMB techniques.

1. Introduction

This document surveys the state of the art of AMB technologies, that is, technologies for Assessment, Measurement and Benchmarking of resilience of computer and information systems. The focus in this survey is on assessment of 'systems' including their unavoidable failures and vulnerabilities, less so on the software development process (which would include discussions about project management, software reliability engineering, etc.) or on proven correctness in design (which would include discussions on formal verification). We discuss methods and techniques related to quantifying the resilience of information systems, through mathematical modelling, measuring, etc.

We interpret 'resilience' assessment to mean assessment of ICT systems in often evolving environments and conditions and in the presence of faults, attacks or failures of any kind. Importantly, we mean to include both security and dependability in the term resilience. We also mean to indicate a broad perspective on modern-day applications for consumers as well as organisations, in which technology must be investigated in unison with its social and economic implications. Please consult Chapter 15 for a further discussion about trends in resilience and the use of terminology in the field.

The individual chapters of this document survey specific AMB technologies, methodologies and standards. In this introduction, we provide a historical perspective on the role, successes, failures and opportunities of AMB technologies in society. This closely relates to an understanding of the role resilience plays in the daily life of computer users and how these users perceive resilience. It also relates to appreciating the role of resilience in business and the legal, regulatory and financial implications for an organisation. We therefore first discuss the role in society of resilience AMB technologies.

1.1. The Role of Resilience and AMB Technologies in Society

Arguably, in many modern-day Internet based systems, resilience takes a back seat. The economic and business case for resilience is apparently insufficiently compelling to allow resilience technologies to compete with functionality and performance. The dominance of first to market business strategies in the Internet age automatically leads to a focus on basic functionality of software as opposed to performance or resilience. One may be tempted to draw the conclusion that there is less value in resilience than in functionality and performance, but we think that this conclusion is too simple. We believe that the limited economic importance of resilience is partly caused by the difficulty in making resilience

'visible' to the end user or the policy maker. That is, the economic and business importance of resilience in computer and information systems would automatically improve if AMB technologies would better succeed in making the consequences of resilience visible. Development of effective AMB technologies would thus lead to business opportunities and economic and societal impact.

To illustrate our point, consider a handheld application (on an iPhone or alike) that finds restaurants on a map of the city one is visiting, utilising GPS and a service that relies on an Internet connection (to obtain maps and other city information). Imagine we would be able to tell the customer when it buys a service that for 'one out of twelve visits to a European city your application will not work', or '3 out of 7 attempts in London will need a retry taking about one minute'. One may expect immediate impact on the consumer's purchase decisions if AMB technologies could provide such information. The handheld application is innocent enough, but given the continued penetration of technology in our society, we expect that the case for resilience needs to be made increasingly often. For instance, if the application is a heart monitor, that sends data to a central hospital, the same poor network connection becomes potentially inacceptable, and customers may demand an assessment of system resilience before purchasing it. The lack of trustworthy AMB technologies therefore threatens the introduction of new Internet services, especially business and safety critical services.

To further illustrate the influence successful resilience AMB technologies can have on society and how they could create business opportunities, it is of interest to consider similar aspects in the area of performance. The mere existence of simple but powerful performance benchmarks steers system designs with respect to performance. For example, the TPC and SPEC performance benchmarks have contributed to improve peak performance of successive generations of systems. AMB technologies in resilience have not been as instrumental in driving technological progress. This is partly for technical reasons: it is much more difficult to assess resilience than it is to assess performance. However, if AMB benchmarks would be as effective as those in performance, it is natural to assume they will be a similar driving force for computer and information system evolution.

The above discussion was motivated by emerging Internet applications, closely associated with the EU's vision of the Future Internet. In contrast to this emerging area, the area of business and safety critical systems routinely conducts resilience assessment, because the consequences of failures are cost of lives or breach of regulations. Resilience is a key element in safety critical systems where failures may cost lives (e.g., aerospace, airplanes, trains). In

such cases the assessment of resilience is driven by government induced regulations. Resilience is often a key element in business critical systems, since the cost of failed services is clear and regulations exist that introduce security as well as dependability requirements. Nevertheless, also in these cases, one could argue that the development of AMB technologies has been lacking and is focused much more on the process of system design than on the resilience of the actual system. Also in such cases, development of AMB technologies is desirable so that the resilience properties of a system become visible through meaningful metrics.

In conclusion, we have argued that the current lack of AMB technologies threatens the further proliferation of Internet services, and that emergence of AMB technologies (such as benchmarks) would dramatically alter the business landscape, creating new commercial incentives for technology progress. We also pointed out that in areas where government regulations enforce AMB techniques to be applied, the emphasis is often on the design and maintenance process instead of the system itself.

1.2. Fundamental Technical Challenges and Historical Milestones for AMB Technologies

Much of the challenges in assessing resilience can be contributed to the nature of faults (and, increasingly important, attacks). Almost all aspects of assessing, measuring or benchmarking a system are complicated through the rare occurrence of faults and failures. This problem of 'rare events' is obvious in both measuring and modelling. In modelling, discrete-event simulation tends to break down because statistical significance can only be obtained by sufficiently often observing the rare event, automatically leading to prohibitively long simulation times. In measurement the problem is identical: one hardly ever observes failures, making it difficult to obtain statistical significance. In this introduction, we provide a high-level overview of AMB technologies covering all chapters of this deliverable, discussed from the perspective of faults and the technical AMB challenges faults introduce.

1.2.1. Measurement

Main Chapters: 7. Fault Injection; 8. Field Measurements; 12. Online Monitoring

Milestones: (see Chapter 7, Section 8.4 and Section 12.1)

- 1970s detection, logging and reacting in hig-end IBM, Tandem and DEC computers
- 1980s Tandem field date published

- 1990s: commercial monitoring tools (HP, IBM) based on SNMP
- 1990s: software implemented fault injection tools
- 2000s autonomic computing initiatives
- 2000s web-based data sharing initiatives

Since the 1980s, there has been important and considerable commercial success in system monitoring, through products such as HP Openview and IBM Tivoli. Important standard such as IETF's SNMP and DMTF's COM guaranteed interoperability, allowing resource data to be collected from any vendor's products. Major industrial players have recently started to point out that monitoring alone is not enough to create resilient computer systems, and have therefore started to push for self-adapting and self-organising systems, in initiatives such as that of IBM's autonomic computing. As a consequence, a flurry of research has emerged in this arena. Industry also was behind the first reports on field data, in particular that of Tandem in the 1980s. Since that period, little field data has been published, impairing the study of system resilience by the academic community. The AMBER data repository is one of the efforts from the research community to improve on this status. System measurement researchers have dealt with the problem of rare fault events through fault injection. Fault injection research has generated a substantial and mature body of research, with considerable progress in a variety of fault injection techniques, both hardware and software implemented, their implementation in tools and initial industrial uptake.

1.2.2. Benchmarking and Certification

Main Chapters: 5. Benchmarking; 11. Safety and Assurance Cases; 13. Robustness Testing

Milestones: (see Figure 5.1, Section 11.1.3, Table 13.1)

- 1970/80s: IT safety guidelines in nuclear and other sectors;
- 1980s: TPC and SPEC performance benchmarks;
- 1990s: software safety cases
- 1990s/2000s: different testing techniques (fault injection, object-oriented
- 2000s: first system dependability and security benchmarks

The category of techniques in this section, have in common the desire to establish the quality of a system with respect to some agreed upon target. The performance community has shown that benchmarks can be a driver for technology. Performance of computer systems is

routinely measured against TPC and SPEC benchmarks, and the results are widely advertised. In the resilience community, benchmarks have been less successful, but the past decade has seen several attempts to improve this situation. In safety-critical systems, a series of important standards have appeared over the years, providing certification criteria for the computing elements in the systems. The research that led to the idea of 'making a case' with sufficient evidence is central to these developments and has lasting impact on the industry. In robustness testing, a number of techniques have been developed in the past two decades such that robustness can be assessed for industrial strength software systems.

1.2.3. Modelling

Main Chapters: 9. Modelling and Model-Based Assessment

Milestones: (see Figure 9.1)

- 1960s: Fault Trees and Petri Nets;
- 1980s: Markov modelling software tools based on Stochastic Petri Nets
- 2000s: integration with model checking and engineering languages such as UML

The problem of rare events has lead to a variety of technological developments in mathematical modelling, setting apart dependability evaluation from performance evaluation. Since discrete-event simulation quickly becomes prohibitively slow (even on modern-day hardware), dependability modelling has led to the successful development of fault tree analysis as well as large scale Markov modelling and solution techniques. Markov modelling techniques were advocated within the framework of performability and tool building has given an important impetus to exploiting Markov modelling. The mentioned techniques are routinely used in hardware and software system design. Currently, important research developments take place in merging solution methods of Markov models with model checking techniques, and in exploiting stochastic process algebras as the higher level modelling language (over the stochastic Petri net variations of the last quarter of the previous century). Also important is the integration of engineering languages (UML, ADL) in such modelling frameworks, to further improve the ability of engineers to apply rigorous modelling techniques.

1.2.4. Shifting Emphasis: Security and Human Factors

Main Chapters: 6. Metrology; 10. Security Assessment; 14. Human Factors; 15. Resilience, Fault Tolerance, Resilience Engineering and its Assessment; 16. Analysis and Presentation Techniques.

Milestones (see Chapter 6, Section 10.4, Section 14.1.1, Chapter 15, Chapter 16)**:**

- 1970s: human reliability analysis
- 1990s: OLAP and other data warehousing tools
- 2000s: emphasis on security: attack injection, honeypots, securitymetrics.org, economics of information security
- 2000s: revisiting base metrology technologies
- 2000s: human error databases
- 2000s: resilience engineering revisits the role of the human in achieving resilience

A shift in emphasis in resilience AMB technologies can be identified. In the domain of human computer interfaces, human reliability analysis techniques of important practical value have been established in the seventies. Recently, the resilience engineering movement has started to rethink the role of humans in the creation and operation of resilient systems. This may lead to important new ways of exploiting human behaviour to improve resilience. Another important shift is the increased importance of security assessment—industry has created an important set of metrics, as witnessed by the active communication on securitymetrics.org. Many challenges still exist to create product oriented metrics, with research activities in honeypots, attack injection and other subareas paving the way.

In addition to these trends in security and human factors, researchers are leveraging successful work from other disciplines and areas. The large foundation of basic techniques and metrology principles, so successfully applied in many other fields, is worth revisiting. Similarly, there is interest to further improve manners of communication to engineers and utilise existing business intelligence and data warehouse tools to analyse large sets of data.

1.3. Contents of this Deliverable

What follows gives more details about the various topics we touched on in this introduction. In addition, Chapter 2 and 3 discuss related European projects and research agendas, and Chapter 4 discusses standardisation efforts.

2. Resilience Assessment in European Projects

The AMBER project is developing a research agenda for resilience assessment as input for the EU Seventh Framework Programme (FP7) of the Information and Communication Technologies (ICT) research activity. It is therefore of relevance to survey the landscape of EU sponsored project that address resilience assessment, and to identify which areas are covered and where gaps still exist or are emerging. We discuss projects in both FP6 and FP7, mostly focussing on ongoing efforts. Certainly, relevant research projects have taken place in earlier programmes, but the results of these projects have found their way into the literature, and are surveyed in the research chapters of this document. Notably is the **DBENCH** project, http://www.laas.fr/DBench/, which was funded under the Fifth Framework Programme. The main goal of DBench project was to devise benchmarks to evaluate and compare the dependability of COTS and COTS-based systems, in embedded, real time, and transactional systems.

2.1. FP6 projects

Within FP6, the predecessor of ICT activity is the Information Society Technologies (IST) area, and most relevant projects are supported through the IST activity. In particular, the sub-programme area 'Towards a Global Dependability and Security Framework' is of relevance. However, other sub-programmes and programmes bring forward projects that intersect with the AMBER interest area as well. We survey the most prominent examples.

DESEREC: Dependability and Security by Enhanced Reconfigurability, www.deserec.eu. The main interest of the proposed DESEREC approach is to improve the dependability by the combination of three technologies: modelling & simulation, incident detection, and response. The work in DESEREC is highly model-driven, including a vulnerabilities and fault model, thus providing a methodology that allows for the assessment of both security and dependability. The project is ongoing and evaluation results will follow.

HIDENETS: Highly Dependable IP-based Networks and Services, www.hidenets.aau.dk. HIDENETS provides end-to-end mobility-aware resilience solutions addressing both accidental and malicious faults, where the user perception of trustworthiness is a key issue. Scenario-based analysis and validation of these solutions is performed via analytic/simulation models, and via an experimental proof-of-concept prototype. Output of HIDENETS (such as the assessment techniques survey) are studied and surveyed in this document.

RESIST: Resilience for Survivability in IST. In the context of ReSIST Network of Excellence, www.resist-noe.eu, an important role is played by the challenges dependability assessment faces. In particular, existing technologies are evaluated taking into account the scaling challenges of large and evolving modern-day systems. State-of-the-art documents from the ReSIST network are used as input for this deliverable.

CRUTIAL: Critical Utility Infrastructure Resilience, http://crutial.cesiricerca.it. In CRUTIAL, assessment was studied in the context of interdependent critical infrastructures in general, and electric power system infrastructures, in particular. The project applies model-based assessment, using discrete-event simulation to deal with difficult to analyse practical fault models including dependencies.

Many other projects provide insights in resilience assessment challenges. For instance, SAFEDMI, http://www.safedmi.org/, deals with railway systems and MEMBRANE, http://www.imperial.ac.uk/membrane/, studies a wireless backhaul network.

2.2. FP7 Coordination Actions

AMBER is one of four Coordination Actions within FP7's Secure, Dependable and Trusted Infrastructure sub-programme area. Only AMBER is concerned with resilience assessment in a broad and encompassing sense, but the other coordination actions touch on some similar aspects. **THINK-TRUST,** http://www.think-trust.eu/**,** discusses social-technical issues of security technologies, **FORWARD,** http://www.ict-forward.eu/, coordinates research around cyber-threats, which undoubtedly will lead it into modelling and assessment questions. **INCO-TRUST,** http://www.inco-trust.eu/, is of a different nature, being an effort to establish coordination across developed countries. For all four coordination actions it holds that they have just started and results still need to be developed, but early dissemination of results among the coordination actions would seem to be of potential benefit for all and for the ensuing research agendas.

2.3. FP7 projects

FP7 has awarded a number of grants to dependability related research. Several of these projects focus on the software development process, an area AMBER does not try to cover: **DEPLOY,** http://www.deploy-project.eu/, on formal techniques, **MANCOOSI**, http://www.mancoosi.org/, on open source, **MOGENTES,** https://www.mogentes.eu/, on testing and **SHIELDS**, http://www.shieldsproject.eu/, on design tool security vulnerabilities. Within the other FP7 dependability projects, assessment almost always plays an important

role, and in many cases assessment needs are discussed at length and novel techniques are announced. Examples of such projects are **GEMOM,** http://www.gemom.eu/, and **SAFT,** http://www.saftbatteries.com/, for middleware dependability, **ACTORS**, http://www.actors-project.eu/, in embedded systems, **AWISSENET,** http://www.awissenet.eu/, in ad-hoc systems, **MASTER,** http://www.master-project.eu/, for business process security assurance, and **INTERSECTION**, http://www.intersection-project.eu/, for integrated networks. A full listing of projects in the sub-programme area of Secure, Dependable and Trusted Infrastructures can be found at http://cordis.europa.eu/fp7/ict/security/projects_en.html.

3. Resilience-related Research Agendas in European Projects

3.1. Topic description

In this chapter we describe research agendas defined in other three previous European projects: AMSD (FP5), GRID (FP6) and RESIST (FP6).

Figure 3.1 depicts a historical timeline of the projects considered and described in this chapter:

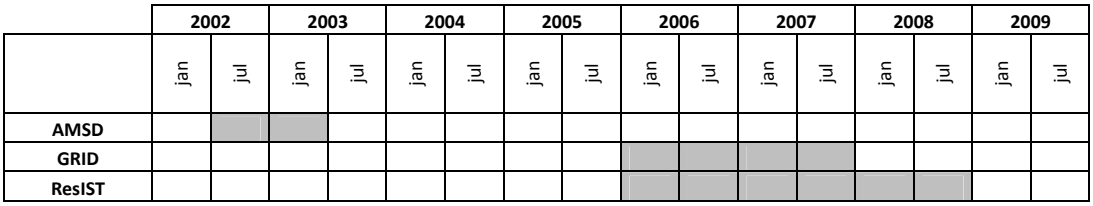


Figure 3.1. Historical time line of projects described in this chapter

3.2. The AMSD project

AMSD (Accompanying Measure System Dependability) was a FP5 "Accompanying and Support Measures" Project, started in June 2002 and ended in May 2003. It addressed the need for a coherent major initiative in FP6 encompassing a full range of dependability-related activities. The results were in terms of an overall dependability road-map that considers dependability in a holistic way and a road-map for dependable embedded systems.

The key objectives of the project were:

- Definition of Overall Dependability Roadmap
- Dependable Embedded System Roadmap
- Constituency and Consensus-Building

The approach of the project is depicted in Figure 3.2; the approach was based on three steps:

– a **scenario analysis**, that departing from future scenarios, aims at obtaining the challenges in need of a solution

– a **roadmap view** on other concurrent roadmaps, that look to what related scientific communities were developing as their vision of future research (looking them through a dependability perspective)

– a **state-of-the-art outlook**, that proceeds from the current state of knowledge and reflects on the themes that indicate further investigation

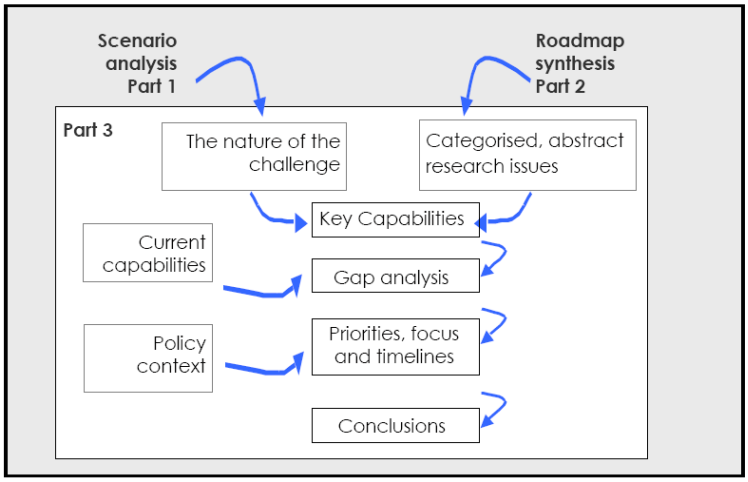


Figure 3.2. Approach used in the AMSD project

3.2.1. Overall dependability roadmap

The following points summarize the characteristics of the overall dependability roadmap defined as result of the AMSD project:

- It aims to cover both technical and socio-technical issues, and a broad range of systems.
- Materials from other more-focused road-maps were consolidated, so as to identify commonalities, tensions, and contradictions as well as opportunities for synergy (see [AMSD3] for details)
- The roadmap used taxonomies/classifications of *dependability* developed by IFIP WG 10.4 (Dependability and Fault Tolerance) as a conceptual framework
- Several target domains were considered: Autonomic, self healing systems; Biometrics; Consumer, network enabled devices; Cryptography; Critical systems; Fault Tolerance; Dependable Embedded Systems; Early Warning and Information Sharing; Evidence/Forensics; Human Factors; Interdependencies (technological and sociological); IT & Law (European and International) ; Large scale bounded systems (enterprise) ; Large scale unbounded (infrastructure); Middleware for critical systems;

Mobile computing, Grid, P2P; Nanotechnology; Privacy and Identity Management; Reliability; Safety; Smart Cards; Survivability; Trust in e-Business and Security

- It identified the **challenges** for **dependability research**, seen under an Information Society prospective light (this part of work is described in [AMSD1])
- It discusses the **gaps** in technology development and examines the **overlaps** with other disciplines
- A set of scenarios were identified [AMSD1]. They were obtained either from existing literature or developed ad-hoc with elements derived from the AMSD workshops. The *micro worlds* described in the scenarios is further developed in a *macro vision,* which considers the *drivers* that shape the landscape where the dependability roadmap will take place. In this way, they look at the different factors that can influence the offer & demand of dependability: technological evolution, application and service markets, changes in business processes and organisations, socio-economic aspects and European/national policies

Key characteristics of the different systems considered in the scenarios are:

1. scale and complexity
2. boundary-less nature of the systems and interconnectedness (few systems have a clear-cut frontier)
3. heterogeneity of human/device boundary (wearable devices, implantable devices, …); etc)
4. incremental development and deployment
5. self-configuration and adaptation
6. multiple innovative kinds of networking architectures
7. multiplicity of fault types, in particular the growing danger of **malicious faults.**

In the project they identified the need to *model* and *understand* multiple interactions and interconnections among systems and they defined a specific *Dependable Embedded Systems.* The main research topics considered were:

- application assessment: the analysis of the different application domains

- technology assessment: the study of the available performance and cost predictions of the enabling technologies, namely semiconductors, communications, dependability, and real-time system and software development
- analysis of the resulting technology/application matrix to identify which fields particularly merit further work, leading establishment of a research agenda.

In AMSD they analyzed several road-mapping reports produced by closely related communities, covering security, mobility and privacy.

The objectives of this work were:

- to take advantage of their efforts
- to consolidate the material from a dependability perspective
- to identify commonalities and potential synergy between research communities.

They thus provided a **dependability-oriented review** that attempts to summarise each roadmap and the challenges.

In **[**AMSD1] they identified challenges of the research activities in the field, which are related to**:**

- The nature of the systems
  - Understanding the **boundary-less nature** of current systems
  - Understanding new risks and threats arising from the **dynamic** and **evolutionary nature** of the systems and their environments
- Understanding and assessing trust, risk and responsibility
  - Predicting **trust relationships** and developing methods for **user–oriented risk assessments**
  - Addressing the regulatory and legal issues
- Developing theories, methods, tools for the design, development and evaluation of systems taking into account
  - The anthropomorphic and autonomous nature of the applications

- The technical problems of scale, non-determinism, and complexity and their dynamic and evolutionary nature

[AMSD4] describes one of results of the project, the identification of the *key dependability capabilities*. It can be summarized as follows:

- To be able to understand, evaluate, communicate, control, remove and mitigate the risks associated with the Ambient Intelligence (AmI) vision
- To understand, evaluate and predict the behaviour of large dynamic (AmI) socio-technical systems of systems
- To understand, evaluate and predict new threats and vulnerabilities
- To be able to design, develop and evaluate cost-effective systems and components that are adequately dependable
- To understand, design, develop and evaluate meta-knowledge and meta-data

Within each capability 5-10 more specific items were elaborated.

In [AMSD4] they also provided a *Gap analysis*; from the identified capabilities they extracted the current position and *how-to* address the gaps. In this deliverable they also identified focus and priorities, which can be summarized as follows:

- Developing a multi-disciplinary dependability community
- Understanding the boundary-less nature of systems and their failure behaviour
- Understanding new risks and threats
- Developing existing dependability technologies to deal with increased scale and complexity and criticality (telecoms, embedded, smart cards)
- Developing theories, methods, tools for the design, development and evaluation of AmI systems
- Understanding and assessing trust, risk and responsibility, predicting trust relationships and developing methods for user – oriented dependability risk assessments

3.2.2. Dependability Research Agenda in AMSD

One of the main objectives of AMSD was to develop a research agenda with both short/medium/long challenges/objectives. Here we summarize the main idea/conclusions behind the development of this research agenda.

A conclusion of the AMSD project was that technology and socio-economic approaches need to work closely together in order to develop a research agenda. Technology will shape the socio-economic settings in which new moral hazard/adverse selection problems will occur but technology will not eliminate the trust/dependability problems. In this project they tried to use the challenges posed by several different future scenarios (in particular in the AmI context) as stimulus for the research agenda. The challenges of the AmI vision requires new forms of interdisciplinarity so that necessary theories / methods / tools needs to be developed.

In particular, the AMSD project produced a general dependability research agenda, with great attention to the field of dependable embedded systems [AMSD4]. They identified three different time horizons for the research topics:

- short-time research (within the next 2 years),
- medium-term research (between 2 and 5 years from now), and
- long-term research (more than 5 years from now),

The top-level synthesis of the R&D challenges of Dependable Embedded Systems is described in [AMSD1].

The ***research agenda*** also contains *timelines* for each key capability. An example of these timelines is depicted in Figure 3.3.

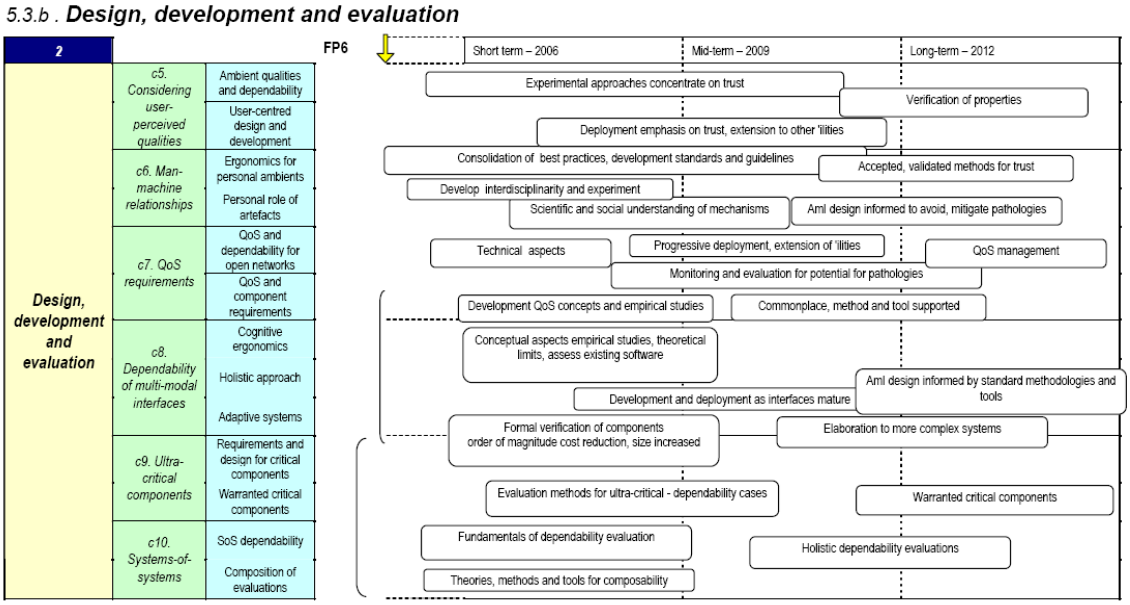


Figure 3.3: Example of timeline in the AMSD research agenda

3.3. The GRID project

GRID was a Coordination Action funded under the Trust and Security objective of the IST Programme of the 6th Framework. The objective of the project was to achieve consensus at the European level on the key issues involved by related to power systems vulnerabilities, in view of the challenges driven by the transformation of the European power infrastructure and ICT integration. The project started in January 2006 and ended in December 2007.

Dealing with power systems vulnerability, the GRID Coordination Action has much to do with resilience. In fact, although not explicitly cited, resilience plays an important role in the objectives the GRID Coordination Action has focused on. This is the reason why it is worth to consider the Research Roadmap established by GRID, described in [GRID].

The GRID project has three main areas for investigation:

- Risk and Vulnerability Assessment Tools and Methods
- Control Architectures and Technologies
- Awareness and Government of Risk in Society

For each of the three pillars, a near term (0-3 years), mid term (3-8 years) and long term (8-15 years) horizons are set. Objectives and research actions are organized accordingly. In the following, resilience-related issues in the GRID research roadmap are presented.

3.3.1. GRID research roadmap

3.3.1.1 Risk and Vulnerability Assessment Tools and Methods

Objectives in the near term are:

1. Identification and understanding of the classes, categories and characteristics of risks and vulnerabilities (present and forecasted)
2. Development of common methodologies for risk assessment and vulnerability analyses of integrated Power and ICT systems

Among the research actions established in the near term, the most interesting from the resilience point of view are:

- Defining the ICT functions and their criticality related to different functional levels of the power system
- Modelling of threat and potential attack categories
- Security metrics and evaluation criteria

Objectives in the mid term are:

1. Development of off-line tools for analyzing the risk and vulnerability related to different hazards and threats (technical, human errors, malicious attacks, weather related, etc)
2. Assessment of Defensive strategies & risks

Among the research actions established in the mid term, the most interesting from the resilience point of view are:

- Modelling of different hazards and threats regarding the physical and logical dimension of power infrastructures
- Investigating methods and tools for "global" vulnerability assessment of the combined power and ICT systems, and their interdependencies
- Integrating human factors (modelling)
- Modelling and simulation tools for assessing the effect of countermeasures

The objective in the long term is the realization of operational tools for vulnerability assessment of components and systems, taking into account expected evolutions and scenarios. This can be accomplished by:

- Adapting the developed tools and methods for real time operation
- ICT-Power Risk-based security operation
- Developing predictive methods for the assessment of the bulk grid trajectory with respect to ICT failure/attack scenarios)
- Integrating human behaviour in critical situations (modelling and laboratory experiments)

3.3.1.2 Control Architectures and Technologies

The near term objective is to establish scientific bases and tools by understanding interdependencies and cascading effects of ICT faults and scenarios. The mid term objectives are :

1. Identification of transition steps towards more robust scenarios
2. Achieve flexible architectures needed to mitigate cascading effects among ICT infrastructures and power systems
3. Development of strategies for decentralized intelligence

The most resilience-related research actions are:

- Investigating control architectures limiting fault propagation “local effects”
- Designing/adapting control architectures allowing operation with degraded modes based on priorities for both power and ICT

The long term objectives are :

1. Assurance of secure supervision & control actions allowing acceptable degraded modes
2. Achieve self reconfiguring architectures and protection mechanisms

The following research actions are outlined to reach those long term objectives:

- Integrating and bridging real time control functions with vulnerability and risk assessment outputs
- Investigating appropriate decision support tools for operators providing real time prospective views on system behaviour in critical states
- Developing self reconfiguring algorithms with optimal solutions from both energy and ICT perspectives with security oriented objective (following threat detection) in order to recover a desired security level
- Developing self adapting and robust protection mechanisms covering the whole control architecture chain with respect to centralized/decentralized control structures

3.3.1.3 Awareness and Government of Risk in Society

On one hand, the demand for appropriate approaches for the management of the associated societal risks and the related education and training processes is peculiar to power systems and for power systems it is particularly critical for power systems, as the European infrastructure consists of many closely interrelated national systems. On the other hand, the need to increase awareness among policy and business circles and technical actors is important also for resilience and resilience measurement, as well as the need to build consensus on risks related to lack of resilience or to the inability of assessing it in a trustable way and to arrive to a standardization of resilience assessment. For these reasons, some of the research actions that the GRID consortium has singled out regarding awareness and government of risk are given in the following:

- Deriving elements to enrich the curricula from case studies, relevant best practices, and outcomes of R&D projects and programmes
- Analysis and proposal of proper risk management structures for companies
- Designing, developing and deploying joint security laboratories, with remote access and information sharing
- Studying and validating secure information exchange protocols, working across companies, across sectors, across countries

- Developing standard formats for the gathering, analysis and dissemination of security relevant information

3.4. The ResIST project

ReSIST was a Network of Excellence (NoE) that addresses the strategic objective "Towards a global dependability and security framework" of the European Union Work Programme, and responds to the stated "need for resilience, self-healing, dynamic content and volatile environments".

The project started in January 2006 and ended in December 2008.

As part of the work, after the overview of the state-of-the-art [ReSIST 06b], a detailed research gap analysis was carried out which was summarized in deliverable "*D13: From Resilience-Building to Resilience-Scaling Technologies: Directions*" [ReSIST 07]. This deliverable contains a detailed list of research gaps identified by the ReSIST partners.

3.4.1. Research gap analysis made in ResIST

3.4.1.1 The process of elaborating the research gaps

The work started with the identification of prospective areas and topics where the different research gaps may exist. This was done in a collaborative way on conferences, brainstorming on meetings and used a web portal (wiki) for sharing the results. Originally each research gap was put into one of the "*assessability*", "*evolvability*", "*usability*" and "*diversity*" categories according to what its initial formulation suggested and then the detailed elaboration started. During this process, different authors from different ReSIST partners added their specific point of view to the description of the gap. This parallel elaboration of the gaps and their continuous evolution resulted in a high quality but very diverse set of research gaps. Thus, at end of the process, an overall analysis had to be made to re-categorize the results into the most appropriate category. In some cases the final gap addressed too many areas and it was split it into different more focused ones. The last step of the whole process was to reveal the relationships between the gaps.

3.4.1.2 Presentation of the research gaps

In order to represent the final results, the material on the collaboration web site was transformed into a continuous document. The following structure was found to best meet the requirements: after introducing the document and its relation to the previous results, the

deliverable virtually consists of two major parts: (i) the synopsis and (ii) the gaps and challenges.

In the first part of the document (i.e., in the synopsis) a few pages long introduction is given on each category, which describes the most important research challenges and gives an overview of the research gaps in the category. In addition, to make the document more comprehensible, the research gaps are further grouped into more focused sub-categories. For example, in the "assessability" category the research gaps are discussed from three perspectives: (i) the impact of the characteristics of current and future *large-scale evolvable systems* on assessability research, (ii) shortcomings of current assessment *methods and techniques*, and (iii) the issue of acceptance of assessment methods and techniques as *industrial engineering methodology*. References are attached to each synopsis. Since the gaps were developed by many collaborators, the list of contributors is represented in the synopsis rather than at the research gaps.

The second part of the document (i.e., the gaps and challenges) contains the research gap descriptions grouped into sections according to the categories; in these sections each gap is described in sequence. The gaps are not prioritized, and the corresponding research actions are not scheduled (no milestones are provided). To keep the flow of train of thoughts consistent, every description is built up from the following parts:

- Definition: a short abstract of the research gap.
- Examples/application domains: real life examples where the research connected to this gap would be important.
- Current approaches: workarounds and techniques to solve corresponding problems.
- Research challenges: what are the real challenges to be solved.
- References: references and bibliography connected to the gap.

3.4.1.3 Research gaps relevant to the aims of the AMBER project

Gaps in the "evolvability" category focus mainly on architectural and design aspects of evolvable resilient systems. These gaps present the following *challenging application areas* for assessing, measuring and benchmarking resilience:

- **Resilient ambient systems:** These are dynamic distributed systems of intelligent objects, where evolvability is essentially the ability of such systems to be resilient to mobility- and failure-induced changes to their composition and/or topology.
- **Trustworthiness/intrusion tolerance in WANs**: Evolvable, scalable trustworthy protocols and systems, like those based on Internet or wide-area networks, are addressed.
- **Resilient data storage**: Storage is considered as a service provided by remote entities, some without central administrative control, where business continuity and disaster recovery requirements have to be fulfilled.
- **Critical infrastructures**: These infrastructures have a level of vulnerability similar to other systems connected to the Internet, but the socio-economic impact of their failures can be huge.
- **Service Oriented Architectures (SOA)**: These are loosely-coupled heterogeneous (typically large-scale) distributed systems where components are given only by their interfaces and are integrated by their functionality.
- **Complexity & self-organisation**: Self-organisation in ubiquitous distributed systems is an underlying mechanism of adaptation.
- **Virtualisation**: It is an architectural approach that allows the real hardware configuration of a system to be abstracted away to provide virtualized storage, networking, and computing resources that can also be used to enhance resilience.

The "assessability" category contains several gaps that are *directly relevant* to the AMBER project as they identify challenges with regard of measuring, benchmarking and assessment of resilience. These challenges are summarized in the following.

- **Data selection, collection, validation**: Empirical data is needed to drive quantitative assessment of resilience. The collection of unbiased, representative and useful data is an open issue. It is particularly evident in the domain of computer security where the lack of data on ongoing attacks does not enable the security actors to identify the most important problems, assess the adequacy of adopted solutions, and compare the various solutions with respect to sound fault assumptions. Research challenges include the issues of definition of relevant data, collection of data, extraction of

relevant information, sharing the data, model building and parameterization on the basis of collected data.

- **Dependability cases**: A dependability case is a structured argument based on assumptions and evidences, which supports a claim that a system meets its specified dependability requirements at a particular level of confidence. Research challenges include claim decomposition, claim-confidence calculus, composition of diverse arguments, including the role of humans, and the evolution of dependability arguments.
- **Security quantification**: The objective is to define quantitative metrics (that can be used to assess the intensity of malicious threats faced by operational systems and the capacity of these systems to resist to potential attacks) and appropriate methods to evaluate them. Research challenges also include the definition of comprehensive models for prediction, definition of realistic assumptions about system vulnerabilities and attacker behaviours, and trade-off analyses between security mechanisms and the ensuing quality-of-service penalties.
- **Benchmarking**: A dependability benchmark is intended first of all to characterize system behaviour in the presence of faults. There is a need for well established and widely accepted dependability benchmarks (approved by recognized consortiums) taking into account important benchmarking properties such as representativeness, repeatability and portability. Since so far only accidental faults were considered in benchmarks, there is a need to develop benchmarks with respect to malicious faults as well. Technical challenges include (i) scaling the current approaches to large-scale systems and infrastructures, (ii) adapting them to evolving systems, and (iii) integration with dependability modelling and experimentation.
- **Metrics/models for evolution processes**: Threats and defences together undergo a continuous process of competitive co-evolution: New threats follow the development of new information systems, and new defences are created and deployed against these threats. The research challenges include (i) the formulation of metrics to evaluate the evolvability of a system, and (ii) the creation of technologies that permit evolvability while ensuring service stability.
- **Model complexity**: Model-based resilience evaluation of increasingly complex systems suffers from the complexity of models that may hinder the solution of the

models. The evaluation of large, dynamic and heterogeneous systems calls for the development of holistic evaluation approaches including complementary evaluation techniques like analytical modelling, simulation and experimental measurements. Evaluation methods shall consider metrics at an increasingly high level of abstraction (to determine the adequate level of detail of the model and abstract the remaining parts of the system). Techniques are needed to quantify the impact of malicious threats and thus provide a comprehensive modelling framework that integrates the evaluation of both accidental faults and malicious threats.

- **Evaluation of dynamic systems**: Applications that feature evolution dimensions such as mobile, ad-hoc based, autonomous, self-organized and ubiquitous applications require specific modelling and evaluation techniques. Holistic evaluation approaches shall be extended with realistic failure models, self-organisation and mobility patterns to support a faithful evaluation.

- **On-line assessment for resilience**: Its goal is the assessment of the implications of the changing environment on system resilience, and thus supporting the necessary adaptation actions. The assessment shall be fast and accurate at the same time, which raises the following challenges: (i) accurate measurements (data collection), (ii) learning and building of (simplified) models from observations, and (iii) assessing the predictive quality of these models in the context of systems operating in a changing environment.

- **Verification of mobile computing systems**: Applications and services on mobile devices may involve device-to-device or device-to infrastructure communication, and they have to be aware of, and adapt to, contextual changes. Research needs include the development of specification and design formalisms, definition of test coverage criteria, model based test generation, and the proper mix of deterministic and probabilistic testing techniques.

- **Compositional reasoning**: It is understood as the deduction of system properties from a collection of sub-properties, and it is applied in order to reduce the complexity of reasoning. Although theoretical approaches exist, there is still a gap to the effective use of compositional verification in practice (e.g., deciding how best to decompose the system or the property, and how to find suitable assumptions that can complete an

assume-guarantee proof). The key research challenge of decomposing dependability claims is strongly related to compositional reasoning.

- **Modelling human behaviour**: Humans are usually considered the critical element of socio-technical systems. There is a need for a conceptual framework providing a complete representation and description of human behaviour within the context of a given environment. There is also the need to understand how the changes in human behaviour influence system resilience.

A research gap in the "usability" category highlights an aspect of resilience that is difficult to measure:

- **Usability metrics**: The challenge in terms of usability metrics is to develop unobtrusive "in the wild" approaches to measuring usability: how to gather the relevant data in context, how the data relates to usability (and thus to resilience), how can the measures be used to compare systems between contexts.

The "diversity" category mainly focuses on architectural and design aspects. Similarly to the "evolvability" group, these gaps identify *challenging application areas* for AMBER technologies, as the effects of diversity have to be measured and assessed. The most relevant cases are the following:

- Diversity for security.
- Large-scale diversity for intrusion tolerance.
- Human diversity and human-machine diversity.
- Reconfiguration and contextual/environmental issues.

3.5. Conclusions

The chapter described research agendas and research gap analysis defined and studied in three previous European projects: AMSD (FP5), GRID (FP6) and ReSIST (FP6). These projects ran in the time range 2002-2009.

AMSD (06/2002-06/2003) attempted at a comprehensive and multi-community elicitation of dependability issues, augmented with the identification of scenarios and statements of the state of the art. It provided a sketch of current capabilities and a gap analysis. In the research agenda they recommended research priorities with specific timelines.

The GRID Coordination Action had much to do with resilience: although not explicitly cited, resilience played an important role in the objectives the project. In the chapter we considered and described the Research Roadmap established by GRID. For each of the main areas for investigation, the research roadmap described a near term (0-3 years), mid term (3-8 years) and long term (8-15 years) horizons, in which objectives and research actions were organized.

The ReSIST NoE (01/2006-12/2008) contributed to the identification of research directions in resilience related research by identifying in 4 categories a set of 41 research gaps that involves more than one hundred research challenges. These cover the design, verification/validation, deployment and operation of resilient computer based systems and networks. Gaps that are relevant to the goals of AMBER and can be taken into account during the elaboration of the AMBER research roadmap were selected and summarized. The gaps identified in the ReSIST project were not integrated into a unified research agenda during the project development.

4. Resilience Assessment Standardisation

4.1. Introduction

Given the fact that software applications are increasingly being used in mission-critical scenarios, there is an increased concern about the conformance of these applications to the specific requirements of quality, safety, dependability and resilience of the application domain. One approach to increase the confidence in the quality of these applications is to mandate that they must adhere to specific standards. These standards are usually tied to the application domain and specify strict requirements that cover many aspects of the development process, such as documentation, audits and proof of safety and dependability.

Standards provide not only guidance for developing dependable and safe systems, but also a way of proving that the development of said applications conforms to a well defined and understood development process, which is assumed to lead to improved quality and resilience of the applications.

Each application domain has its own set of specific quality and resilience requirement. Thus, there are currently many standards for application development for specific application domains. Examples of standards that directly mention and provide dependability requirements include: IEC 61508 safety standard for all kinds of industry, ESA's ECSS standards such as ECSS-E-40 and ECSS-E-70-41A for space software, UK MoD Def Stan 00-55 and 05-95 (UK) for safety-critical software for defence equipment, CENELEC EN-50126 and 50128 for railway, RTCA/DO-178B for the aeronautics industry, ISO 26262 for the automotive industry, IEC 62304:2006 for the life cycle requirements for medical device software, and BSI BS IEC 60880 for software requirements for nuclear power plants.

Given the large diversity of application areas and corresponding standards, we decided to look at three examples of standards for software applications to indicate how dependability and resilience assessments aspects are addressed in current standards:

- The EN 50126, 50128, 50129, 50159, 61508 set of standards for Railway Systems. These standards address the specification, the design and the validation of dependability-related aspects concerning railway systems and applications, including Reliability, Availability, Maintainability and Safety in control, electronic signalling, and transmission systems, among others. These standards were issued

in the last decade by the CENELEC (Comité Européen de Normalisation Electrotechnique).

- The ISO WD 26262 Standard for automotive systems. The ISO WD 26262, baseline 11 is a working draft of an upcoming standard for ensuring functional safety of electrical and electronic systems in road vehicles. It is applicable to safety-related electrical and electronic systems, including programmable systems, used in road vehicles, and is prepared by Technical Committee ISO/TC 22, Road vehicles, Subcommittee SC 3, Electrical and electronic equipment. The new standard is expected to become available in 2010.
- The ISO/IEC 25051 for COTS software products. This standard specifies the documents that must accompany COTS software products and was prepared by Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and system engineering. The standard specifies the quality requirements for COTS software, the requirements for test documentation, including test cases and test reports, and instructions for conformity evaluation.

4.2. Railway Systems

In the last decade the CENELEC (Comité Européen de Normalisation Electrotechnique) has issued some standards regarding the specification, the design and the validation of dependability-related aspects concerning railway systems and applications. As well known, CENELEC standards have to be translated without any modification and applied by all the member countries.

The most important European standards in the field are the:

- EN 50126 “Railway Applications: The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)”, issued in October 1998;
- EN 50128 “Railway Applications: Communications, Signalling and Processing Systems – Software for railway control and protection systems”, issued in 2000;
- EN 50129 “Railway Applications - Communication, signalling and processing systems: Safety related electronic systems for signalling”, issued in 2001;
- EN 61508 “Functional safety of electrical/electronic/programmable electronic safety-related systems” issued in 2002;
- EN 50159-1 “Railway Applications: Communications, Signalling and Processing Systems – Part 1: Safety-related communication in closed transmission systems”, issued in 2001;

- EN 50159-2 “Railway Applications: Communications, Signalling and Processing Systems – Part 2: Safety-related communication in open transmission systems”, issued in 2001.

It is worth noting that these standards do not explicitly consider the assessment of resilience or of typical dependability attributes, but rather deal with some aspects that can be considered as pertinent to dependability, such as Reliability, Availability, Maintainability and, above all, Safety (RAMS). Moreover, they do not prescribe the use of one or more techniques to perform the assessment, but suggest/prescribe methodologies and identify operative steps and documentation that should be prepared and analyzed by the assessor.

Among the standards listed above, the two most relevant are for sure the EN 50126 and the EN 50129, and therefore the aspects of these standards that can be of interest for the chapter are separately dealt with in the following. The standard EN 50128, which basically focuses on methods and instruments to *develop* software systems development is also cited, as the annexes C, D and E of its part 5 (EN 61508-5) give some interesting qualitative and quantitative examples on the determination of safety integrity.

4.2.1. Standard EN 50126

As stated in the Scope section, among other objectives, the standard EN 50126 defines railway RAMS and defines “a process, based on system life cycle and tasks within it for managing RAMS”, but it does not define “rules or processes pertaining to the certification of railway products against the requirements of the[1] standard”.

When defining railway RAMS, the standard also investigates the factors that can influence it (sec. 4.4 of the standard). These factors are divided into three categories: sources of failures introduced within the system (system conditions), imposed on the system during operations (operations conditions), and imposed on the system during maintenance (maintenance conditions). The standard states that factors that could influence the RAMS need to be identified, their effect assessed, and the cause of these effects managed throughout the lifecycle of the system.

Moreover, it prescribes the specification of RAMS requirements and of a RAMS programme, involving the definition of RAM failure categories and the analysis of their associated *risk*, to be performed at various phases of the system life cycle and adequately documented. Probability or frequency of occurrence of hazardous events, as well as hazard severity level connected with each identified hazard should be analyzed and presented, for

[1] In this sentence the article *the* replaces the adjective *this* which is used in the standard.

example, in terms of a frequency/consequence matrix, which makes it easy to evaluate the risk and separate among different risk levels (e.g. undesirable, intolerable, tolerable, negligible). Once the level of safety of the application has been determined and the risk estimated, safety integrity requirements can be derived.

The standard defines the system lifecycle as a sequence of phases covering the total life of the system. The lifecycle provides a structure for planning, managing, controlling and monitoring all aspects of the system including RAMS. The phases are:

1. Concept
2. System definition and application conditions
3. Risk analysis
4. System requirements
5. Apportionment of system requirements
6. Design and implementation
7. Manufacturing
8. Installation
9. System Acceptance
10. Operation and Maintenance
11. Performance Monitoring
12. Decommissioning and Disposal

For each phase, the standard identifies phase-related RAM and Safety tasks.

In order to facilitate the application of the standard, the annexes give some interesting examples. Note that the annexes are only *informative*, i.e. they are not *normative*.

Annex A gives a principal outline of RAMS specifications for rolling stock, identifying operating and environmental conditions, reliability targets, system failure modes and mean time between failures, maintenance and repair requirements (e.g. mean time to repair), safety targets, hazardous conditions, safety-related functions and failures, safety hazards severity levels, system availability.

Annex B gives an outline procedure for a basic RAMS programme and lists some appropriate methods and tools for conducting and managing a RAMS programme. In particular, it refers to some procedures for performing deductive (top-down) and inductive (bottom-up) preliminary, worst case and in-depth RAMS analysis, which are included in separate standards. The most relevant procedures listed are: diagnostic testing, failure modes and effect analysis (FMEA), fault tree analysis (FTA), reliability block diagram method and

Markov techniques. Regarding safety, the standard cites and invites to refer also to the standard EN 61508 which will be presented in one of the next subsections.

4.2.2. Standard EN 50129

The standard EN 50129 is "the first European Standard defining requirements for the acceptance and approval of safety-related electronic systems in the railway signalling field". It includes both hardware and software aspects.

There are three conditions for safety acceptance:

- evidence of quality management;
- evidence of safety management;
- evidence of functional and technical safety.

A structured safety justification document (the *Safety Case*) must be redacted and must carry the documentary evidence of quality management, safety management and functional and technical safety. It is a fundamental part of the documentation to be given to the safety authority in order to obtain safety approval.

The quality management system shall be applicable throughout the system/sub-system/equipment life-cycle, as defined in EN 50126.

Satisfying the second condition for safety acceptance means that the safety of the system should be managed by means of an effective safety management process, consistent with the management process for RAMS reported in EN 50126. The standard EN 50129 states that "safety management process shall be implemented under the control of an appropriate safety organisation, using competent personnel assigned to specific roles". Moreover, at the beginning of the life cycle, a Safety Plan should be written to identify the safety management structure, safety-related activities and approval milestones throughout the life cycle.

Regarding the third condition, the technical evidence for the safety of the design must be documented in the Technical Safety Report, which is a part of the Safety Case. The Technical Safety Report must explain the technical principles which assure the safety of the design. It should include all supporting evidence (e.g. design principles and calculations, test specifications and results, and safety analyses). Its structure is detailed in Annex B of the standard, which is *normative*. The Technical Safety Report must consist of six sections:

1. Introduction;
2. Assurance of correct functional operation;

3. Effects of faults;
4. Operation with external influences;
5. Safety-related application conditions;
6. Safety Qualification tests.

The most interesting, with regard to *resilience*, is section 3. This section concerns "the ability of the system to continue to meet its specified safety requirements in the event of random hardware faults and, as far as reasonably practicable, systematic faults". Particular aspects that must be considered are:

- Effects of single faults;
- Independence of items;
- Detection of single faults;
- Action following detection (including retention of safe state);
- Effects of multiple faults;
- Defence against systematic faults.

For example, regarding the *Effects of single faults*, it must be ensured that the THR (Tolerable Hazard Rate) is met in the event of a single random fault. "It is necessary to ensure that SIL 3 and SIL 4 systems remain safe in the event of any kind of single random hardware fault which is recognised as possible". Annex B also gives some examples of how to make the system *resilient*[2] in case of single faults, but gives no operative specification on how to assess its resilience.

The standard also prescribes that an *independent* safety assessment of the system and its Safety Case should be carried out before an application for Safety approval can be considered. The results of this assessment, whose methodologies are not imposed by the standard, should be presented in a Safety Assessment Report.

It is the Railways Authority's responsibility to perform risk analysis, including the hazard identification, consequence analysis, risk estimation and the tolerable hazard rates (THR) allocation. It is the supplier's responsibility to perform the hazard analysis, including causal analysis and SIL (Safety Integrity Level) allocation. Hazard analysis involves both an empirical phase, based on past experience (e.g. checklists) and a creative phase of proactive

[2] Note that the term *resilient* is not explicitly used in the standard, although the concept of resilient underlies section 3 of the Technical Safety Report as defined by the standard.

forecasting (e.g. brainstorming, what-if studies). In some cases, risk analysis and hazard analysis are not completely independent. The hazard analysis can lead to system changes which offer more safety performance. Both the risk analysis and the hazard analysis need the approval of the safety authority.

The Annex A of the standard, which is *normative*, assign each SIL level (from 1 to 4) a THR per hour and per function (e.g. the maximum SIL, which is 4, implies $10^{-9} \leq THR^3 < 10^{-8)}$.

4.2.3. Standard EN 61508

The text of the standard IEC 61508 (1998) was approved by CENELEC as a European Standard without any modification as EN 61508. It relates to the functional safety of electrical/electronic/programmable electronic safety-related systems and consists of 7 parts.

In the Introduction section of standard EN 50129 it is stated that "This standard and EN 50126 are based on the system life-cycle and are in line with IEC 61508, which is replaced by the set of EN 50126/50128/50129, as far as Railway Communication, Signalling and Processing Systems are involved. Meeting the requirements in these standards is sufficient to ensure that further compliance to IEC 61508 need not be evaluated". However, it is interesting to observe that Annexes C, D and E of part 5 (EN 61508-5) give some interesting qualitative and quantitative examples on the determination of safety integrity levels. In particular, in Annex C the average probability of failure is quantitatively determined for a safety-related protection system, whereas in Annex D and Annex E two qualitative methods, respectively based on a risk graph and a hazardous event severity matrix are presented.

4.3. Automotive systems

This section gives an overview of ISO WD 26262, baseline 11, a working draft of an upcoming standard for ensuring functional safety of electrical and electronic systems in road vehicles. The ISO WD 26262 is an adaptation of the well-known IEC 61508 standard to the automotive sector developed by the International Organization for Standardization (ISO). It is applicable to safety-related electrical and electronic systems, including programmable systems, used in road vehicles, and is prepared by Technical Committee ISO/TC 22, Road vehicles, Subcommittee SC 3, Electrical and electronic equipment. The new standard is expected to become available in 2010.

[3] THR in the formula is to be intended *per hour and per function*.

The draft standard defines functional safety as "Absence of unacceptable risk due to hazards caused by mal-functional behaviour of E/E systems" (E/E means electrical and/or electronic). It provides requirements and processes for mitigating risks that arise from systematic faults in hardware and software, as well as random faults in hardware. The main features include:

- An automotive safety lifecycle model covering management, development, production, operation, service and decommissioning of systems.
- A definition of risk classes known as Automotive Safety Integrity Levels, or ASILs.
- A methodology for specifying safety requirements based on ASILs; and
- Requirements for "validation and confirmation measures" for ensuring safety.

A reference process model based on the well known V-model is used as a basis for the draft standard, which consists of nine parts. Each part is described in a separate document. Part 3, 4, 5, 6 and 7 of the standard deal with different phases of the V-model. The remaining parts address glossary (Part 1), management (Part 2), supporting processes (Part 8) and safety analyses (Part 9). In the following we describe the different parts of the draft standard, highlighting references to techniques that are of relevance for AMBER, i.e., techniques for assessment, benchmarking and measurement of safety.

4.3.1. ISO WD 26262 Part 1 – Glossary

The glossary defines 126 terms, which are essential to the application of the standard. A complete description of these terms is beyond the scope of this survey. In Table 4.1 we give the definitions of some terms that are of general interest. (In these definitions, the word *item* refers to a system or a function to which the standard is applied.)

Table 4.1 Definition of selected terms from ISO WD 26262

Term	Definition
Assessment	Independent examination of product characteristics.
Audit	Independent examination of process implementation.
Automotive Safety Integrity Level (ASIL)	One of four classes to specify the item's necessary safety requirements for achieving an acceptable level of residual risk with A representing the lowest and D the highest class.

Confirmation measure	Review, audit or assessment concerning functional safety.
Field data	Data logged during previous usage of an item or element including cumulated operating hours, all failures and in service problems.
Verification	Determination of completeness and correct specification or implementation of requirements from a previous phase.
Formal verification	Mathematical proof of an algorithm or a specification against properties.
Informal verification	Verification techniques not regarded as semi-formal or formal verification techniques.
Safety goal	Top-level safety requirement as a result of the hazard analysis
Safety validation	Confirmation based on examination and tests that the safety goals are sufficient and have been achieved.
Testing	Process of planning, preparing and executing or exercising a system or system component to verify that it satisfies specified requirements, to detect errors and to create confidence.
Black-box testing	Tests that do not use knowledge of the internal structure or implementation of the test object.
White-box testing	Tests where knowledge of the internal structure and implementation of the test object is used.

As we can see in Table 4.1, the glossary defines several general terms such as *testing*, *verification*, *validation* and *assessment*. The definitions of these words are similar but not identical to those given in the widely used taxonomy developed by [Avizienis 04]. The definitions provided in ISO WD 26262 tend to be specific than those given by Avizienis et al. For example, the standard defines only the specialized term *safety validation*, whereas Avizienis et al. use validation as a general term meaning "checking the specification". It is interesting to note that the standard defines *assessment* as an "independent examination of product characteristics", and thus reserves this word for activities carried out by an independent body.

4.3.2. ISO WD 26262 Part 2 – Management of functional safety

Part 2 of the draft standard presents a safety lifecycle model that constitutes a basis for the entire standard. The safety lifecycle model covers three phases: *concept phase*, *product development*, and *"after start of production"*. The model represents an ideal development

process, which allows loops and iterations although these are not explicitly shown in the model. Activities carried out during the concept phase are described in Part 3 and include, for example, *initiation of the safety lifecycle*, *hazard analysis* and *risk assessment*.

The product development phase includes activities related to *product development*, *operation planning* and *production planning*. The product development activities are divided into three levels of development activities: *system level*, *hardware level* and *software level*, which are described by Part 4, 5 and 6, respectively. Activities related to operation planning and to production planning are described in Part 7.

The elements of Part 2 discussed so far are described in a section called "Overall project independent safety assessment". This section also discusses issues such as safety culture, quality management, training and qualification of personnel.

In a section called "Project-dependent safety management during development", requirements on "confirmation measures for ensuring functional safety" are given. They state that the following activities should be performed during the development process:

a) Audit of safety related processes;
b) Review of work products (reports, etc) of safety activities;
c) Assessment of functional safety.

The technical scope of AMBER is mainly concerned with activities related to the last item.

4.3.3. Part 3 – ISO WD 26262 Concept phase

This phase is divided into four sub-phases: *Item definition*, *Initiation of the safety lifecycle*, *Hazard analysis and risk assessment*, and *Functional safety concept*. The first sub-phase aims at defining and describing an item, including its functional specification and the way it interacts with other items and the environment. The second sub-phase involves determining whether the item is a new development or a modification of an item, and tailoring the safety life cycle and the corresponding activities that should be carried out to ensure safety. The third sub-phase aims to identify hazards and define safety goals concerning prevention and mitigation of these hazards so that the residual risk becomes acceptable. The fourth sub-phase involves deriving safety requirements based on the safety goals established during the previous sub-phase.

One of the key activities during the concept phase is to determine ASIL-levels for the product under development. This takes place during the hazard analysis and risk assessment

sub-phase. As already mentioned, there are four ASIL levels denoted A to D, where A corresponds to the lowest safety level and D corresponds to the highest safety level. In addition, there is also a class QM (quality management) used to denote that an assignment of an ASIL level is not required.

The activities performed during the concept phase do not include safety assessment, but the ASIL assignment determines the types of assessment activities that are to be conducted during the product development phases.

4.3.4. Part 4 – ISO WD 26262 Product development: system level

This part describes the activities that take place during product development at the system level. This phase consists of two groups of sub-phases. The first group consists of three sub-phases: *initiation of product development at the system level*, *specification of the technical safety concept* and *system design*. They precede the hardware design activities described in Part 5, and the software design activities described in Part 6. The second group of sub-phases follows after the hardware and software development activities. These sub-phases are: *item integration and testing*; *safety validation*; *functional safety assessment*; and *product release*. We focus our attention on the first three sub-phases in the second group, as they are strongly related to the technical scope of AMBER.

Item integration and testing

For the *item integration and testing* sub-phase, the draft standard provides lists of test methods that are applicable in different situations. One example is the list of test methods recommended for software-hardware integration. It consists of ten test methods: 1) Requirement-based tests, 2) Back-to-back tests, 3) Test of external interfaces, 4) Interface consistency check, 5) Tests of internal interfaces, 6) Fault injection tests, 7) Error guessing tests, 8) Resources usage tests, 9) Performance tests, and 10) Stress test.

Recommendations regarding the use of these tests are given for each ASIL level. A test method is designated either as *recommended* or *highly recommended* for a given ASIL level. For example, fault injection is recommended for level A and B, and highly recommended for level C and D. The draft standard requires that a detailed rationale must be provided if a developer decides not to use a highly recommended method. The use of a recommended method is optional.

Safety validation

For the *safety validation* sub-phase, the draft standard requires that a detailed validation plan is developed. It states that the following methods and measures for validation shall be used:

- Repeatable tests, e.g., black-box, simulation, fault injection, stress tests and simulation of external influences;
- Analyses, e.g., FMEA, FTA, ETA and simulation;
- Long term tests, user tests under real-life conditions;
- Test with persons, expert panels; and
- Reviews.

Functional safety

The purpose of the *functional safety assessment* sub-phase is to provide a recommendation of whether an item shall be granted a *qualified acceptance*, an *acceptance* or a *rejection*. An agenda for the assessment of functional safety must be prepared in this sub-phase. Such an agenda contains a list of outcomes of design and verification activities which constitute the input to the assessment. An example of an agenda for assessment of functional safety for ASIL D items given in the draft standard includes the following points: 1) Systems functions, 2) Hardware, 3) Safety concept, 4) Safety analysis and safety data, 5) Safety design process for the lifecycle phases, 6) Software development, 7) Verification and test, 8) Documentation and safety case, and 9) Summary and assessment. Hence, the functional safety assessment is a high level activity that involves an assessment of the outcomes of many design and verification activities.

4.3.5. Part 5 – ISO WD 26262 Product development: Hardware level

This part of the draft standard contains seven sub-phases covering activities related to hardware design: 1) Initiation of product development at the hardware level; 2) Specification of hardware safety requirements; 3) Hardware design; 4) Hardware architectural constraints; 5) Assessment criteria for probability of violation of safety goals; 6) Hardware integration and testing; and 7) Safety requirements for hardware software interface. Sub-phases 4, 5 and 6 address issues related to safety assessment and validation.

Hardware architectural constraints

The objective of the *hardware architectural constraints* sub-phase is "to evaluate the hardware architecture of the item against requirements of fault handling as represented by the

safety architectural metrics". To this end, the draft standard defines two "safety architectural metrics" for characterizing the coverage of safety mechanisms: the *single point fault metric* and the *latent fault metric*. These metrics are applicable only to random hardware faults and are essentially coverage factors, i.e., probabilities that the safety mechanisms successfully prevent a hardware fault from causing a hazardous event.

Assessment of criteria for probability of violation of safety goals

The *assessment of criteria for probability of violation of safety goals* (sub-phase 5) involves demonstrating that the probability for random hardware failures to cause safety violations is sufficiently low. The standard proposes two methods for achieving this objective. One is based on a probabilistic analysis involving estimation of failure rates for all hardware components in the system. The other involves a prediction of occurrence rates of hardware failures that constitute single and double points of failure.

Hardware integration and testing

For *hardware integration and testing* (sub-phase 6), the standard proposes a large number of recommendations for how to perform safety integration testing for different ASIL levels. Among the recommended testing methods are various functional tests, fault insertion and accelerated life tests.

4.3.6. Part 6 - ISO WD 26262 Product development: Software level

Part 6 consists of seven sub-phases including three "top-down" design phases and three "bottom-up" testing phases. In addition, it includes a sub-phase for starting up the software development activities. Thus, the software product development phase consists of the following sub-phases: 1) Initiation of product development at the software level; 2) Specification of software safety requirements; 3) Software architectural design; 4) Software unit design and implementation; 5) Software unit testing; 6) Software integration and testing; and 7) Software safety acceptance testing. Safety validation and assessment activities are mainly carried out in the last three sub-phases

Software unit testing, and software integration and testing

For *software unit testing* (sub-phase 5), the standard contains tables of recommended methods and measures for software unit testing, functional unit testing and for determining structural test cost coverage. Similar tables are also provided for the *software integration and testing* (sub-phase 6). Among the test methods mentioned are *requirements based tests*, *external interface tests*, *fault injection tests* and *error guessing tests*. Requirements based test

are highly recommended for all ASIL categories, while external interface tests are highly recommended for ASIL categories B, C and D. Fault injection and error guessing tests are highly recommended for ASIL categories C and D.

Software safety acceptance testing

For *software safety acceptance testing* (sub-phase 7), five categories of tests are recommended or highly recommended: *tests of software safety requirements, use of test interface* (for monitoring the internal state of electronic control units during test), *hardware-in-the-loop-test*, *tests within the electronic control unit network*, and *tests in the test vehicle*. (These are obviously categories of tests rather than specific test methods, but they are still called test methods and measures in the standard.)

4.3.7. Part 7 - ISO WD 26262 Production and operation

This part provides requirements and guidelines for ensuring and maintaining the safety of a vehicle and its subsystems during production and operation. This is the shortest part of the standard, comprising only 10 pages. (Other parts range from 18 to 53 pages.) It contains two lists of requirements (called provisions), one for vehicle production and one for vehicle operation. The operation phase includes use of the vehicle, service and decommissioning. (Service includes repair and maintenance.) The work products (outputs) of this phase are all documents.

4.3.8. Part 8 - ISO WD 26262 Supporting processes

Part 8 provides requirements and guidelines for ten supporting processes: 1) Interfaces within distributed development, 2) Overall management of safety requirements, 3) Configuration management, 4) Change management, 5) Verification, 6) Documentation, 7) Qualification of software tools, 8) Qualification of software components, 9) Qualification of hardware components, and 10) Proven in use argument.

The verification process is a generic process which is instantiated by sub-phases of Part 3, 4, 5, 6 and 7. The draft standard uses the term verification in a very broad sense. In the concept phase (part 3), verification means to evaluate the concepts to ensure that they are feasible to implement and correct with respect to the boundary conditions of the system. During development, verification is conducted both during the design phases and the testing phases. In the design phases, verification refers to the evaluation of the work products such as requirements or architecture specifications through reviews, simulation or analysis techniques. In the test phases, verification means to ensure that the system or subsystem developed

complies with the requirements. Examples of verification methods and measures mentioned in the standard include *model-checking*, *simulation*, *testing*, *engineering analyses*, *demonstration*, *review*, *walkthrough* and *inspection*.

We here briefly describe the processes for *verification*, *qualification of software components*, *qualification of hardware components*, and for *employing proven-in-use arguments*, as these processes are the ones most relevant for AMBER.

Verification

A long list of provisions (requirements) is stated for the verification process. This list is divided into provisions for three groups of activities: 1) Planning of verification, 2) Specification of verification, and 3) Execution of verification.

Planning of verification is required for all phases and sub-phases of the safety life-cycle. It shall consider issues such as selection of verification tools, criteria for acceptance/rejection of the work product under verification, planning of resources and schedules of verification activities, and regression strategy.

Qualification of software components

The *qualification of software components* sub-phase aims to demonstrate that software components are suitable for use in safety-related systems. This involves providing evidence that the components fulfil requirements with respect to a diverse set of properties. The draft standard demands developers to consider the following properties whenever appropriate: 1) *functionality*; 2) *algorithmic accuracy*; 3) *numerical accuracy*; 4) *reliability*; 5) *behaviour in case of defect*; 6) *response time*; 7) *resource usage*; 8) *behaviour in overload situation*; 9) *data security*; and 10) *clear and unambiguous interfaces*.

Clearly, several of these properties can be viewed as *resilience* properties, even though the standard does not explicitly identify them as such. In fact, no attempt is made to group properties into different categories. Thus, resilience properties are mixed with timing properties, performance properties and functional properties.

Qualification of hardware components

The objective of the hardware qualification sub-phase is to produce arguments that a hardware component fulfils its safety requirements during normal use and when exposed to environmental stress. The process applies to hardware parts that make up a subsystem such as an ECU, a sensor or an actuator. It is not intended for qualification of off-the-shelf hardware

parts such as integrated circuits, as the standard points to other established procedures for qualification of such parts.

The standard identifies three methods that be used for qualification of hardware components: *proven in use*, *testing* and *analyses*. Qualification can be done using a single method or a combination of these methods.

Proven in use argument

This process gives guidelines for how a proven in use argument should be developed. Proven in use argument is an alternate means of demonstrating that a system, subsystem or component fulfils a given ASIL requirement.

A proven in use argument consists of extensive documentation of the item, including change management, configuration control and field data. Employing a proven in use argument is possible, for example, when an existing function or system is carried over from one vehicle to another, or when an item has been developed according to other safety standards than ISO 26262, such as IEC 61508 or RTCA DO 178B, and has a recorded usage history.

4.3.9. Part 9 - ISO WD 26262 Supporting processes

This part of the standard describes four analysis techniques: 1) ASIL decomposition, 2) Criticality analysis, 3) Analysis of dependent failures and 4) Safety analyses.

ASIL decomposition is a method for mapping the ASIL level initially assigned to an item onto the architectural elements which implement the item. (The standard defines an item as an array of systems, a system or a function, and an element as a system, subsystem or component.) The method provides a set of rules for how to assign ASIL requirements to redundant elements, which together implement an item with a certain ASIL requirement. *Criticality analysis* is a method for determining ASIL requirements for internal elements of an item. In general, all elements of an item inherit the ASIL requirement assigned to the item. The method aims at identifying possibilities of relaxing the ASIL requirement for a subset of the elements that make up an item, without changing the ASIL requirement of the item itself. To reduce the ASIL of an element, the analysis must demonstrate that potential systematic faults in the element cannot violate the safety requirements of the item.

Analysis of dependent failures aims at identifying situations where an item may fail, or violate its safety requirements, due to a lack of independence among the elements that make of the item. Possible sources of such failures include ineffective handling of systematic faults and lack of fault/error containment.

Finally, the section on *safety analysis* discusses the use of several established method for analysing the impact of faults on system safety. Both qualitative methods, such as Failure Mode Effects Analysis (FMEA), Fault Tree Analysis (FTA), Event Tree Analysis (ETA) and Hazard and Operability Studies (HAZOP), as well as quantitative methods, such as Markov modelling and reliability block diagrams, are discussed.

4.4. COTS Software Products

Commercial Off-The-Shelf (COTS) software products are currently being used in an increasingly wide variety of application including mission-critical scenarios with strict requirements for quality and resilience. Thus, the correct operation of COTS is often vital for business, safety or personal applications.

COTS software products are ready-made packages, often of a generic-purpose nature, sold off-the-shelf to the acquirer who had no influence on its features and qualities. One driving force behind the widespread use of COTS is the fact that it is cheaper and faster to obtain pre-existing COTS products than to develop them in-house. This however brings into the development cycle an unknown factor of risk.

Typically, the documentation provided with the COTS product is the only means whereby the acquirer can gather information on the specification and intrinsic qualities of the product. It is therefore important that essential information is given to enable acquirers to evaluate the quality of the COTS software products for their needs.

The SO/IEC 25051 [SO/IEC 25051] standard is the Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) — Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing. This standard addresses the documentation that must accompany the COTS software product and also addresses the testing procedure of the software product. It was prepared by Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and system engineering. This edition cancels and replaces the first edition of ISO/IEC 12119:1994.

This standard is generic and by COTS software product is addressable by it. Examples include but are not limited to text processors, spreadsheets, data base control software, graphics packages, software for technical, scientific or real-time embedded functions, automated teller machines, and web software such as generators of web sites.

This standard addresses only the user confidence that the COTS software product will perform as offered and delivered. It does not deal with the production process. The quality system of a supplier is outside the scope of this standard.

This Standard specifies:

a) Quality requirements for COTS software products;
b) Requirements for test documentation for the testing of COTS software products, including test requirements, test cases, and test reporting;

4.4.1. Users

The intended users of this International Standard include:

1. suppliers, when:
    a) specifying requirements for a COTS software product;
    b) advertising performance claims for their product;
    c) assessing their own software products against the claimed performance;
    d) issuing declarations of conformity;
    e) applying for certificates or marks of conformity;
2. certification bodies that may wish to establish a third-party certification scheme
3. testing laboratories
4. accreditation bodies for accrediting registration or certification bodies and testing laboratories;
5. potential acquirers who may:
    a) compare the requirements for the intended work task with the information in product descriptions of existing software products;
    b) look for certified COTS software product;
    c) check if the requirements are otherwise met;
6. end users who may profit from better software products;
7. organizations:
    a) establishing management and engineering environments based on the quality requirements and methods of this international standard;
    b) managing and improving their quality processes and personnel;
8. regulatory authorities who may require or recommend the requirements of this International Standard for COTS software products used in safety or business-critical applications.

4.4.2. Conformance

A COTS software product conforms to this International Standard if:

1. it has the properties and requirements specified in the standard;

2. it has been tested by producing test documentation that meets the requirements of Clause 6;
3. anomalies found during testing are documented and resolved prior to product release. Anomalies against advertised performance claims must be fixed or the performance claim must be removed. Known anomalies may be considered acceptable if:
    a) the anomaly is not a violation of a performance claim; and
    b) the supplier has duly considered the nature and the impact of the anomaly on the potential acquirer and deemed it negligible, and has preserved the documentation of the anomalies for future improvement.

4.4.3. Requirements

The requirements are organized in the following sections

- Product description. This includes:
    - Availability: the product description shall be available for potential acquirers and users of the product;
    - Contents: the product description shall contain information needed by potential acquirers to evaluate the suitability of the software for their needs, and the product description shall be free from internal inconsistencies. These claims must be verifiable.
    - Identification and indications: The product shall display a unique identification and have a name, version and date. It must have the identification and address of the manufacturer. It must identify the intended uses and indicate the applicable law and regulations. It must describe interfaces, external dependencies and concurrency restrictions. It must specify any dependency on maintenance.
    - Statements of functionality. The product description shall contain, as applicable, statements on functionality, taking into account suitability, accuracy, interoperability, security, and functionality compliance, written such that verifiable evidence of compliance can be demonstrated. It must provide an overview of any external callable functions and offer a description of its most critical functions. All known limitations to user functionality shall be described.
    - Statements of reliability. The product description shall contain, as applicable, statements on reliability, taking into account maturity, fault tolerance, recoverability, and reliability compliance, written such that verifiable evidence of compliance can be demonstrated. The product description shall address the ability of the software to continue operating in the case of user interface errors, errors in the application's own logic, or errors due to availability of system or network resources. The product description shall include information on data saving and restoring procedures.
    - Statements of usability. The product description shall contain, as applicable, statements on usability, taking into account understandability, learnability, operability, attractiveness, and usability compliance, written

such that verifiable evidence of compliance can be demonstrated. Any specify skill required to use the product must be identified.

- Statements on efficiency,. The product description shall include statements of efficiency, taking into account time behaviour, resource utilization, and efficiency compliance, written such that verifiable evidence of compliance can be demonstrated.
- Statements on maintainability. The product description shall contain, as applicable, statements on maintainability, taking into account analyzability, changeability, stability, testability, and maintainability compliance, written such that verifiable evidence of compliance can be demonstrated,
- Statements on portability. The product description shall contain, as applicable, statements on portability, taking into account adaptability, installability, replaceability, co-existence, and portability compliance, written such that verifiable evidence of compliance can be demonstrated. The product description shall specify the different configurations or supported configurations (hardware, software) for putting the software into use.

- User documentation. The user documentation must be complete (The user documentation shall contain the information necessary for the use of the software and shall describe all the functions stated in the product description and all functions that the end user can call.), must be correct and unambiguous (all information must be traceable to its authorative source of correctness), must be consistent and free of contradictions, be understandable by the end user population.
- Quality requirements for software. These address the functionality (all functions mentioned in the user documentation shall be executable with the corresponding facilities, properties, and data, and within the limitations given there. The functions of the software shall be executable according to all the statements in the user documentation), reliability (the software must perform in accordance with the reliability features defined in the user documentation, and the function related to error handling shall be consistent with corresponding statements in the product description and in the user documentation), usability (the questions, messages, and results of the software execution shall be understandable), efficiency, and maintainability.
- Test documentation. The content requirements are a) the test plan, b) the test description, and c) the tests results. The test documentation shall contain a list of all the documents that compose it, with their titles and their identifiers. Each document of the test documentation shall include:
  - a title;
  - a single identifier (reference, number of version, date of issue).
  - a history of the modifications or any other element describing the evolution of the document;
  - contents or a description of the content;
  - the identifier of the documents referred to in the body of the document;
  - information relating to the authors and the inspectors;

- a glossary.

- Test description. No specific test techniques or methods are recommended. Each quality characteristics mentioned in the product description shall be the objective of at least one test case. Also, all the functions described in the user documentation, as well as the combinations of functions representative of the task to be achieved, shall be subject to test cases. The test cases shall demonstrate the conformity of the software to the statements in the user documentation. The level of functional decomposition selected as basis for the test case design shall be indicated. All the installation procedures shall be subject to test cases. All the operational limits indicated in the product description and user documentation shall be subject to test cases. The criteria used to decide if the test results demonstrate the conformity of the software to the product description and user documentation shall be indicated. The test case description shall include:

  a) its test objective;

  b) a unique identifier;

  c) input data and test boundaries for test;

  d) the detailed steps to perform;

  e) the expected behaviour of the system;

  f) the expected output from the test case;

  g) the criteria for the result interpretation;

  h) the criteria used to decide on positive or negative result of the test case.

  The test procedure shall include:

  a) the test preparation;

  b) the actions necessary to begin and to execute the test;

  c) the actions necessary to record the test results;

  d) the conditions and actions to stop and eventually restart the tests.

4.4.4. Conformity evaluation

The standard specifies instruction for the evaluation of conformity. This can be summarized as follow:

- Presence of the COTS elements: All components of all the computer systems as described in the product description shall exist and be available for conformity evaluation;
- Product description conformity evaluation: A conformity evaluation is carried out to determine the conformity of the product description to the requirements of product description (availability, contents, identification and indications, statements of functionality, known limitations, usability, etc.);
- Conformity of the test description including test results and corrections made;

The standard allows for third party conformity evaluation. The process is as follows:

- The supplier provides the COTS software product to the third-party. The supplier can also provide test documentation.
- If the supplier provides only the COTS software product, without the test documentation, the third-party shall:
    a) carry out a conformity evaluation of the product description, the user documentation, and the software;
    b) record the results in a conformity evaluation report.
- If the supplier provides the COTS software product and the test documentation, the third-party shall:
    a) carry out a conformity evaluation of the product description and the user documentation;
    b) carry out a conformity evaluation to determine the conformity of the test documentation;
    c) record the results in a conformity evaluation report.
- The conformity evaluation report is composed by:
    a) the COTS software product identification;
    b) the date of evaluation completion and, if any, testing completion;
    c) if any, the computer systems used for testing (hardware, software, and their configuration);
    d) the documents used, with their identification;
    e) the summary of conformity evaluation activities and, if any, testing activities;
    f) the summary of conformity evaluation results and, if any, testing results;
    g) the detailed results of conformity evaluation and, if any, testing;
    h) if any, the list of non-conformities to requirements.

4.5. Conclusions

Failures in railway systems can have catastrophic consequences including loss of life. Thus, it is a domain for which standards exist to ensure that these systems comply with a given set of quality requirements. The most important European standards for Railway systems and applications are the EN 50126, which addresses the specification and demonstration of reliability, availability, maintainability and safety (RAMS), the EN 50128 which addresses communications, signalling and processing systems for software for railway control and protection systems, the EN 50129 addressing communication, signalling and processing systems for safety related electronic systems for signalling, the EN 61508 which addresses functional safety of electrical/electronic/programmable electronic safety-related

systems, and the EN 50159 addressing communications, Signalling and Processing Systems for safety-related communication in transmission systems.

The ISO WD 26262 draft standard provides a comprehensive framework for ensuring the safety of electrical, electronic and programmable automotive systems, covering the entire life-cycle from requirements elicitation to decommissioning. It is unique compared to other dependability or safety oriented standards in that it provides detailed requirements on how to assess, validate, and verify designs and implementations at the system, software and hardware levels. It also includes a comprehensive glossary, as well as requirements and guidelines for management, supporting processes and safety analyses.

COTS software products offer and attractive cost-efficient way of building systems. However, they represent the inclusion on unknown risk factors into the larger system. Standards offer a coherent and uniform method to describe and to assess the quality and resilience of COTS that can help do decide whether to use or not a given COTS product. The SO/IEC 2501 addresses the documentation that must accompany the product as well as the specification of the testing procedures for that product. This standard is generic and in theory any COTS software product is addressable.

Many of these standards use or require technologies addressed by AMBER, such as fault injection, analytical modelling, and analysis of field data. By explicitly endorsing the use of these technologies, the ISO WD 26262 is likely to become an important driver for the development of resilience and safety assessment techniques during the next decade.

5. Benchmarking

Computer benchmarks are standard tools that allow evaluating and comparing different systems or components according to specific characteristics (performance, dependability, security, etc). To achieve acceptance by the computer industry or by the user community a benchmark should fulfil a set of key benchmarking properties: representativeness, portability, repeatability, scalability, non-intrusiveness, and simplicity of use. These properties must be taken into account from the beginning of the definition of the benchmark components and must be validated after the benchmark has been completely defined.

The work on performance benchmarking has started long ago. Ranging from simple benchmarks that target a very specific hardware system or component to very complex benchmarks focusing complex systems (e.g., database management systems, operating systems), performance benchmarks have contributed to improve successive generations of systems. Research on dependability benchmarking boosted in the beginning of the present decade, having already led to the proposal of several dependability benchmarks. Several works have been carried out by different groups and following different approaches (e.g., experimental, modelling, fault injection). Due to the increasing relevance of security aspects, security benchmarking is becoming a new player in the benchmarking field. However, research on security benchmarking is still in an embryonic stage. Figure 5.1 presents an overview of the evolution of benchmarking over the last four decades.

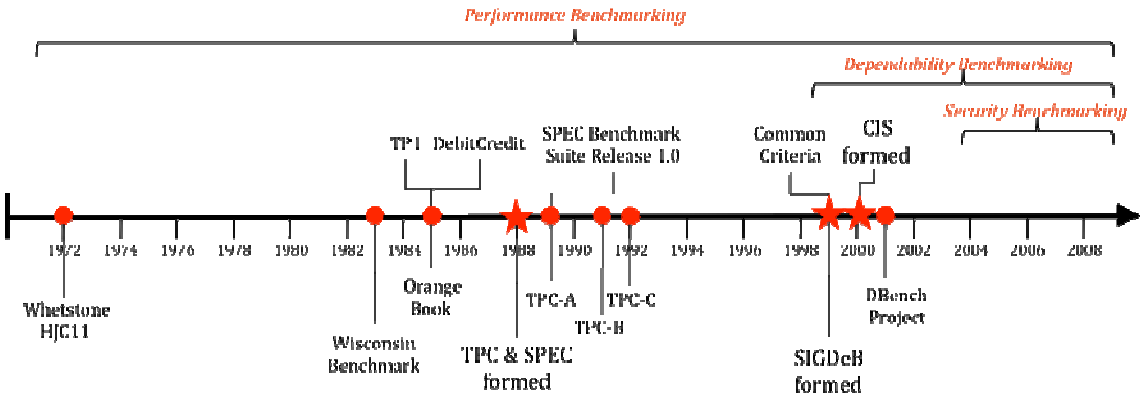


Figure 5.1. Overview of benchmarking history.

Whetstone programs, the first general-purpose benchmarks that set industry standards for computer system performance, appeared almost four decades ago with the HJC11 benchmark [Curnow 76], which was written in Algol 60 in 1972 at the National Physical Laboratory in the United Kingdom. Whetstone is a very well known synthetic benchmark for evaluating the

performance of computers in terms of instructions executed per second.

Several performance benchmarking initiatives were launched in the 1970s and 1980s. The Wisconsin benchmark [Bitton 83] was launched in 1983 and marked the beginning of a new generation of performance benchmarking: benchmarking of complex database systems. In fact, the work on performance benchmarking of database systems (and transactional systems in general) is large, and two years after the Wisconsin benchmark, the TP1 and Gray's DebitCredit [Anon 85] benchmarks were launched. These benchmarks settled the foundations for the Transaction Processing Performance Council (TPC) performance benchmarking initiative, which started in 1988 when the TPC organization was formed [TPC].

The first TPC benchmark, TPC-A [TPC-A 94] (a benchmark to measure performance in update-intensive database environments typical in on-line transaction processing applications), was released in 1989. TPC-B [TPC-B 94] (a benchmark to measure throughput in terms of how many transactions per second a system can perform) was the second TPC benchmark and was launched one year after TPC-A. The most successful TPC benchmark, TPC-C [TPC-C 07] (an on-line transaction processing benchmark that measures transactions per minute and cost per transaction) was released in 1992. TPC-C is still today the reference benchmark for complex database environments (although it went through many revisions).

The major milestone on dependability benchmarking was the creation of the "Special Interest Group on Dependability Benchmarking" (SIGDeB) [SIGDeB] by the International Federation for Information Processing (IFIP) Working Group 10.4 [IFIPWG10.4] in 1999. The goal of this group is to promote the research, practice, and adoption of dependability benchmarks. The DBench project [DBench], started in 2001, was funded by the European Commission, under the Information Society Technologies Programme (IST), Fifth Framework Programme (FP5). DBench was the first major project on dependability benchmarking and settled the foundations for many other initiatives. Several dependability benchmarks were defined in the context of the DBench project, targeting the following application domains: embedded, real time, and transactional systems.

Although security benchmarking is a new research topic, work on security assessment started long ago. One of the main works is the Orange Book [DoD 85], provided by the US Department of Defense in 1985, with the goal of defining a set of generic rules to specify and evaluate security attributes of software systems. Another major contribution is the Common Criteria for Information Technology Security Evaluation [CCriteria 99], made available in 1999. The Center for Internet Security (CIS) [CIS] is a non-profit organization, formed in

2000 with the goal of creating a series of security configuration assessment documents for several commercial and open source systems.

This chapter presents an overview on the state-of-the-art on benchmarking performance, dependability and security. The goal is to identify the current approaches, techniques and problems relevant to the resilience benchmarking problem.

5.1. Performance benchmarking

Performance benchmarks are standard procedures and tools aiming at evaluating and comparing different systems or components according to specific performance measures. The two most prominent organizations working on the performance benchmarking business are TPC (Transaction Processing Performance Council) [TPC] and SPEC (Standard Performance Evaluation Corporation) [SPEC].

A performance benchmark is typically provided as a computer program ready to be executed (as is the case of SPEC benchmarks) or as a document that specifies how the benchmark should be implemented and run (this is the case of TPC benchmarks) and is normally composed by three main elements: workload, measures, and rules. The workload defines the set of tasks that the system under benchmarking has to perform during the benchmark execution. The measures portray the time needed to execute the workload or, alternatively, the number of operations executed per unit of time. The purpose of the rules is to assure that the results from different executions of the benchmark are valid and comparable.

The **Transaction Processing Performance Council (TPC)** is a non-profit organization whose goal is to define and disseminate benchmarks for databases and transactional systems in general. TPC also verifies the results announced by benchmark performers in order to make those results official. TPC is composed by the major vendors of systems and software from the transaction processing and database market. Detailed information on TPC organization and benchmarks can be obtained at the TPC web-site [TPC].

TPC has currently four benchmarks: TPC-C [TPC-C 07] and TPC-E [TPC-E 08] for OLTP (On-Line Transaction Processing) systems, TPC-App [TPC-App 05] for application servers and web services, and TPC-H [TPC-H 08] for decision support systems. These benchmarks measure transaction processing and system performance in terms of how many operations or transactions a given system can execute per unit of time.

The **Standard Performance Evaluation Corporation (SPEC)** is a non-profit organization

that establishes and maintains a set of benchmarks in several domains. SPEC develops suites of benchmarks and reviews and publishes submitted results. SPEC is an organization made of three different groups (detailed and updated information on SPEC organization, groups, and benchmarks, can be obtained from the SPEC web-site [SPEC]):

- The Open Systems Group (OSG) is the oldest group within SPEC. The OSG focus is on component-level and systems-level benchmarks for desktop systems, workstations and servers running open operating system environments.
- The goal of the High Performance Group (HPG) is to establish, maintain and support a set of performance benchmarks that target high performance system architectures, such as symmetric multiprocessor systems, workstation clusters, distributed memory parallel systems, and traditional vector and vector parallel supercomputers.
- The Graphics & Workstation Performance Group (PEC/GWPG) is the umbrella organization for autonomous groups that develop graphics and workstation benchmarks and performance reporting procedures.

SPEC benchmarks consist of programs taken from real-life applications and modified to improve portability and fulfil some specific requirements of performance benchmarking [Eigenmann 01]. Typically, to run a SPEC benchmark the user only needs to compile the source code for a specific environment.

5.2. Dependability Benchmarking

The awareness of the importance of dependability benchmarks has increased in the recent years and dependability benchmarking is currently the subject of intense research. A dependability benchmark is a specification of a standard procedure to measure dependability attributes of computer systems or components. The main goal is to provide a standardized way to compare different systems or components from a dependability point-of-view.

Comparing to typical performance benchmarks, such as the TPC and SPEC benchmarks, which consist mainly of a workload and a set of performance measures, dependability benchmarks typically add two new elements: 1) measures related to dependability; and 2) a faultload that emulates real faults experienced by systems in the field.

The following subsections present some recent advances on dependability benchmarking, both at universities and computer industry sites.

5.2.1. Special Interest Group on Dependability Benchmarking (SIGDeB)

The Special Interest Group on Dependability Benchmarking (SIGDeB) [SIGDeB] was created by the International Federation for Information Processing (IFIP) Working Group 10.4 [IFIPWG10.4] in 1999 to promote the research, practice, and adoption of benchmarks for computer-related systems dependability. The work carried out in the context of the SIGDeB is particularly relevant and merges contributions from both industry and academia.

A preliminary proposal issued by the SIGDeB was in the form of a set of standardized classes for characterizing the dependability of computer systems [Wilson 02]. The goal of the proposed classification was to allow the comparison among computer systems concerning four different dimensions: availability, data integrity, disaster recovery, and security. The authors have specifically developed the details of the proposal for transaction processing applications. This work proposes that the evaluation of a system should be done by answering a set of standardized questions or performing tests that validate the evaluation criteria.

A very relevant effort in the context of SIGDeB is a book on dependability benchmarking of computer systems [Kanoun 08]. This book presents several relevant benchmarking initiatives carried out by different organizations, ranging from academia to large industrial companies.

5.2.2. DBench Project

The DBench project [DBench] was funded by the European Commission, under the Information Society Technologies Programme (IST), Fifth Framework Programme (FP5). The main goal of DBench project was to devise benchmarks to evaluate and compare the dependability of COTS and COTS-based systems, in embedded, real time, and transactional systems. Several works on dependability benchmarking have been carried out in the DBench project. The following subsections summarize those works.

5.2.2.1 General purpose operating systems

The works presented in [Kalakech 04a, Kalakech 04b, Kanoun 05, Kanoun 06] address the problem of dependability benchmarking for general purpose operating systems (OS), focusing mainly on the robustness of the OS (in particular of the OS kernel) with respect to faulty applications.

The measures provided are: 1) OS robustness in the presence of faulty system calls; 2) OS reaction time for faulty system calls; and 3) OS restart time after the activation of faulty system calls. Three workloads are considered: 1) a realistic application that implements the experiments control system of the TPC-C performance benchmark [TPC-C 07]; 2) the

PostMark [Katcher 97] file system performance benchmark for operating systems; and 3) the Java Virtual Machine (JVM) middleware. The faultload is based on the corruption of systems call parameters.

Another research work on the practical characterization of operating systems behaviour in the presence of software faults in OS components is presented in [Durães 02]. The methodology used is based on the emulation of software faults in device drivers and the observation of the behaviour of the overall system regarding a comprehensive set of failure modes analyzed according to different dimensions related to different user perspectives.

5.2.2.2 Real time kernels in onboard space systems

The work presented in [Moreira 03] is a preliminary proposal of a dependability benchmark for real time kernels for onboard space systems. This benchmark, called DBench-RTK, focuses mainly on the assessment of the predictability of response time of service calls in a Real-Time Kernel (RTK).

The DBench-RTK dependability benchmark provides a single measure that represents the predictability of response time of the service calls of RTKs used in space domain systems. The workload consists in an Onboard Scheduler (OBS) process based on a functional model derived from the Packet Utilization Standard [PUS 03]. The faultload consists of a set of faults that are injected into kernel functions calls at the parameter level by corrupting parameter values.

5.2.2.3 Engine control applications in automotive systems

The work presented in [Ruiz 04] represents a preliminary proposal of a dependability benchmark for engine control applications for automotive systems. This benchmark focuses on the robustness of the control applications running inside the Electronic Control Units (ECU) with respect to transient hardware faults.

This dependability benchmark provides a set of measures that allows the comparison of the safety of different engine control systems. The workload is based on the standards used in Europe for the emission certification of light duty vehicles [EECDirective 99]. The faultload consists of transient hardware faults that affect the cells of the memory holding the software used in the engine control.

5.2.2.4 On-line transaction processing systems

The DBench-OLTP dependability benchmark [Vieira 03a, Vieira 03b] is a dependability

benchmark for on-line transaction processing systems. The DBench-OLTP measures are divided in three groups: baseline performance measures, performance measures in the presence of the faultload, and dependability measures. The DBench-OLTP benchmark can be used considering three different faultloads each one based on a different class of faults, namely: operator faults, software faults and high-level hardware failures.

In [Buchacker 03] it is presented a preliminary proposal of another dependability benchmark for on-line transaction processing systems. The measures provided by this dependability benchmark are the system availability and the total cost of failures. These measures are based on both measurements obtained from experimentation (e.g., percentages of the various failure modes) and external data (e.g., the failure rates and the repair rates). The external data used to calculate the measures must be provided by the benchmark user. The workload was adopted from the TPC-C performance benchmark [TPC-C 07] and the faultload includes exclusively hardware faults, such as faults in the storage hardware and in the network.

5.2.2.5 Web-servers

[Durães 04b] proposes a dependability benchmark for web-servers (the WEB-DB dependability benchmark). This dependability benchmark uses the basic experimental setup, the workload, and the performance measures specified in the SPECWeb99 performance benchmark [SPECWeb 99].

The measures reported by WEB-DB are grouped into three categories: baseline performance measures, performance measures in the presence of the faultload, and dependability measures. The WEB-DB benchmark uses two different faultloads: one based on software faults that emulate realistic software defects (see [Durães 04a]) and another based on operational faults that emulate the effects of hardware and operator faults.

5.2.3. Berkeley University

The work developed at Berkeley University has highly contributed to the progress of research on dependability benchmarking in the last few years, principally on what concerns benchmarking the dependability of human-assisted recovery processes.

[Brown 00] introduces a general methodology for benchmarking the availability of computer systems. The workload and performance measures are adopted from existing performance benchmarks and the measure of availability of the system under test is defined in terms of the service provided by the system. The faultload (called fault workload by the authors) may be composed of a single-fault (single-fault workload) or of several faults (multi-fault workload).

[Brown 01] addresses human error as an important aspect in system dependability, and proposes that human behaviour must be considered in dependability benchmarks and system designs. [Brown 02] proposes a technique to develop dependability benchmarks that capture the impact of human operators on the tested system. The workload and measures are adopted from existing performance benchmarks and the dependability of the system can be characterized by examining how the performance measures deviate from their normal values as the system is perturbed by injected faults. In addition to faults injected using traditional fault injection, perturbations are generated by actions of human operators that actually participate in the benchmarking procedure.

[Brown 04a] presents the first steps towards the development of a dependability benchmark for human assisted recovery processes and tools. This work proposes a methodology to evaluate human-assisted failure recovery tools and processes in server systems. This methodology can be used to both quantify the dependability of single recovery systems and compare different recovery approaches, and combines dependability benchmarking with human user studies.

5.2.4. Carnegie Mellon University

Vajra [Narasimhan 08] is a research project whose goal is benchmarking the survivability in distributed systems, focusing on the objective and quantitative comparison of the runtime implementations of different Byzantine fault-tolerant distributed systems. The benchmark uses as the point of injection APIs that are common across various Byzantine fault-tolerant systems. A variety of accidental and malicious faults are injected at various rates across the system.

Although not resulting in a formal benchmark proposal, the research on robustness testing [Koopman 00] developed at the Carnegie Mellon University has effectively set the basis for robustness benchmarks of operating systems. This will be further discussed in Chapter 13, devoted to the survey of robustness testing techniques.

5.2.5. Sun Microsystems

Research at Sun Microsystems has defined a high-level framework [Zhu 03a] specifically dedicated to availability benchmarking of computer systems. The proposed framework follows a hierarchical approach that decomposes availability into three key components: fault/maintenance rate, robustness, and recovery. The goal was to develop a suite of benchmarks, each one measuring an aspect of the availability of the system. Within the

framework proposed by [Zhu 03a], two specific benchmarks have already been developed.

[Zhu 03b] proposes a benchmark for measuring a system's robustness (degree of protection that exists in a system against outage events) in handling maintenance events, such as the replacement of a failed hardware component or the installation of a software patch.

In [Mauro 04] it is proposed a benchmark for measuring system recovery in a non-clustered standalone system. This benchmark measures three specific system events; clean system shutdown (provides a baseline metric), clean system bootstrap (corresponds to rebooting a system following a clean shutdown), and a system reboot after a fatal fault event (provides a metric that represents the time between the injection of a fault and the moment when the system returns to a useful state).

Another effort at Sun Microsystems is the Analytical RAS Benchmarks [Elling 08], which consists of three analytical benchmarks that examine the Reliability, Availability, and Serviceability (RAS) characteristics of computer systems:

- The Fault Robustness Benchmark (FRB-A) allows assessing and comparing the techniques used to enhance resiliency, including redundancy and automatic fault correction.
- The Maintenance Robustness Benchmark (MRB-A) assesses how maintenance activities affect the ability of the system to provide a continuous service.
- The Service Complexity Benchmark (SCB-A) examines the complexity of mechanical components replacement.

5.2.6. Intel Corporation

Work at Intel Corporation has focused on benchmarking semiconductor technology. [Constantinescu 05a] shows the impact of semiconductor technology scaling on neutron induced SER (soft error rate) and presents an experimental methodology and results of accelerated measurements carried out on Intel Itanium® microprocessors. The proposed approach can be used as a dependability benchmarking tool and does not require proprietary information about the microprocessor under benchmarking.

Another study [Constantinescu 05b] presents a set of benchmarks that rely on environmental test tools to benchmark undetected computational errors, also known as silent data corruption (SDC). In this work, a temperature and voltage operating test (known as the four corners test) is performed on several prototype systems.

5.2.7. IBM Autonomic Computing Initiative

At IBM, the Autonomic Computing initiative [AutonomicIBM] is developing benchmarks to quantify a system's level of autonomic capability, which is defined as the capacity of the system to react autonomously to problems and changes in the environment. The goal is to produce a suite of benchmarks covering the four categories of autonomic capabilities: self-configuration, self-healing, self-optimization, and self-protection.

[Lightstone 03] describes the first steps towards a benchmark for autonomic computing. The benchmark addresses the four attributes of autonomic computing and is able to test systems at different levels of autonomic maturity.

[Brown 04b] identifies the challenges and pitfalls that must be taken into account in the development of benchmarks for autonomic computing capabilities. This paper proposes the use of the workload and driver system from performance benchmarks and the introduction of changes into benchmarking environment in order to characterize a given autonomic capability of the system. [Brown 04b] proposes that autonomic benchmarks must quantify the level of the response, the quality of the response, the impact of the response on the users, and the cost of any extra resources needed to support the autonomic response.

5.3. Security Benchmarking

Several security evaluation methods have been proposed in the past [CCriteria 99, DoD 85, CEC 93, RedTeam]. The Orange Book [DoD 85] and the Common Criteria for Information Technology Security Evaluation [CCriteria 99] define a set of generic rules that allow developers to specify the security attributes of their products and evaluators to verify if products actually meet their claims. Another example is the red team strategy [RedTeam], which consists of a group of experts trying to hack its own computer systems to evaluate security. To the best of our knowledge, none of these security evaluation approaches is oriented towards security benchmarking, as comparing security has been largely absent from these security evaluation methods.

The work presented in [Maxion 00] addresses the problem of determining, in a thorough and consistent way, the reliability and accuracy of anomaly detectors. This work addresses some key aspects that must be taken into consideration when benchmarking the performance of anomaly detection in the cyber-domain.

The set of security configuration benchmarks created by the Center for Internet Security (CIS) is a very interesting initiative [CIS]. CIS is a non-profit organization formed by several well-

known academic, commercial, and governmental entities that has created a series of security configuration documents for several commercial and open source systems. These documents focus on the practical aspects of the configuration of these systems and state the concrete values each configuration option should have in order to enhance overall security of real installations. Although CIS refers to these documents as benchmarks they mainly reflect best practices and are not explicitly designed for systems assessment or comparison.

[Vieira 05] proposes a practical way to characterize the security mechanisms in database systems. In this approach database management systems (DBMS) are classified according to a set of security classes ranging from Class 0 to Class 5 (from the worst to the best). Systems are classified in a given class according to the security requirements satisfied.

In [Neto 08] the authors analyze the security best practices behind the many configuration options available in several well-known DBMS. These security best practices are then generalized and used to define a set of configuration tests that can be used to compare different database installations.

5.4. Open Research Challenges

This chapter presented the state-of-the-art on benchmarking. The work on performance benchmarking has started long ago and has contributed to improve successive generations of systems. Dependability benchmarking efforts both at universities and computer industry sites are quite recent. Security is a newcomer to the benchmarking world and little work has been performed so far.

Although performance benchmarking is a very well established field, further work on dependability benchmarking seems to be necessary in several application areas (e.g., real-time systems, grid computing, parallel systems, etc). Additionally, no dependability benchmark has achieved the status of a real benchmark endorsed by a standardization body. This may be due to several reasons (that need to be studied) but clearly shows that additional work is still needed.

In the area of security benchmarking, a lot of work is clearly needed, as this is a new and quite challenging field for which few work has been developed so far. A key issue is the definition of useful and meaningful security *metrics*. In fact, the problem of security quantification is a longstanding one. A useful security metric must portray the degree to which security goals are met in a given system, allowing a system administrator to make informed decisions. One of the biggest difficulties in designing such a metric is related to the fact that security is, usually,

much more dependent on what is *unknown* about the system than on what is known about it. In fact, security metrics are hard to define and compute as they involve making isolated estimations about the ability of an unknown individual (e.g., a hacker) to discover and maliciously exploit an unknown system characteristic (e.g., a vulnerability).

6. Metrology

The past years have seen a growing interest in methods for studying the behaviour of computer-based systems. The scientific literature as well as the industrial practice shows that measuring resilience and dependability attributes is a key issue: in particular, experimental measurement is an attractive option for evaluating an existing system or prototype, because it allows observing the real execution of the system in its real environment to obtain highly accurate measurements of the system.

In several contexts we need to observe the behaviour of computing systems, mainly:

- to monitor their actual behaviour (e.g. to on-line react to events);
- to evaluate their behaviour (e.g. to quantitatively assess a system).

In both these contexts the more precise we are in the observation of the system, the more reliable are the results we collect, and more effective are the decisions we are able to take:

- in on-line monitoring (see the chapter dedicated to on-line monitorig), we are able to react in a more appropriate way;
- in the evaluation, we are able to take decisions (off-line), using more precisely collected information.

Tools for experimental assessment and monitoring of resilience and dependability properties should be treated for what they really are: measurement instruments. Methodology and tools for evaluation and monitoring of distributed systems can thus successfully use the conceptual framework and mathematical tools and techniques made available by metrology, the science devoted to studying the measuring instruments and the processes of measuring. First of all, since measuring a quantity (the measurand) consists in quantitatively characterizing it, a clear and univocal definition of the measurands is of uttermost importance. Metrology has developed theories and good practice rules to make measurements, to evaluate measurements results and to characterize measuring instruments. The main metrological properties (uncertainty, repeatability, resolution, sensitivity and intrusiveness) should be precisely identified in each methodology (and supporting tool) for experimental assessment and monitoring of resilience and dependability properties of computing system. In a similar way, the results obtained using a tool should include uncertainty evaluation and, when comparing results achieved through different measurement methods, compatibility of measurement results should be assessed.

Dependability and resilience measurements on computing systems involve a wide variety of measures, from *discrete* measures, such as number of source code lines, packet size in packet-switched networks, to *continuous* measures that refer to the dynamic behaviour of the system under evaluation, which include delays experienced in an end-to-end connection, quality of clock synchronization, quality of service metrics, etc. A closer look at this class highlights the crucial role of time measurements: dependability-related measurements are very often based on time measurements, for example because the measurand is a time interval, or because the measurement result is obtained through indirect measurements based on timestamps.

In this chapter we investigate the awareness and correct application of the main metrological concepts in literature when designing tools and experiments for the assessment of *dependability* and *resilience* properties.

A complete presentation of the basic concepts and definitions in metrology science can be found in [BIMP 08].

Issues with the way measurement is applied in assessing computer dependability, and the need for giving practice a better theoretical basis, were first raised with respect to software reliability assessment. Problems were identified separately in two communities of research and practice: software reliability [Brocklehurst 92] and software metrics [Fenton 94]. There were three sets of inter-related issues: confusion about the meaning of a measure (leading for instance to redefining software "reliability" as a count of bugs in a piece of code, or to seeking scalar measures for inherently multi-dimensional attributes), confusion between problems of measurement and of prediction (leading for instance to naïve methods for inference from observed failures to future reliability), and insufficient fitness for purpose of the metrics [Fenton 97].

A more recent paper [Bondavalli 07a] raised the problem of awareness of measurement theory in evaluating dependability attributes of computing systems. In the paper, a set of well-known tools for experimental assessment of dependability and papers describing results of experimental evaluations are analyzed, identifying whether and to what extent the most important metrological properties and attributes, namely *uncertainty*, *repeatability*, *resolution*, *intrusiveness* and results *compatibility*, are taken into account. Up to now, it is the only document that presents a deep state-of-the-art in this area.

Considering the metrology research community, some works on the performance assessment of distributed measurement systems and computer networks, mainly in terms of the evaluation

of some specific network parameters (e.g. one-way delay, packet loss ratio, jitter) have been published. Some papers concerning fault diagnosis in electronic and automotive systems and some others related to the design of distributed measurement systems for specific applications, such as power quality measurements, are also available in the literature. Nevertheless, the problem of measuring and assessing resilience of computing systems with a strict and systematic metrological approach has never been faced.

Computing systems are not typically designed to be monitored; a methodological approach for their observation is thus needed. In distributed systems things are even more complex, for the lack of central control, and for the difficulties in obtaining a precise global time and an accurate view of the global state of the system. Dependability issues are very rarely addressed in the major conferences and scientific journals in the area, and resilience is practically not considered at all.

In Section 6.1 we focus on the main results in the dependability community regarding the problem of awareness and applications of measurement theory; results of this section greatly overlap with the ones presented in [Bondavalli 07a]. In Section 6.2 some papers involving dependability and resilience issues, included in the major Instrumentation and Measurement conference proceedings and journals are enlisted, and the way dependability and resilience are defined and studied is discussed. In Section 6.3 we survey the main open research challenges of this area.

6.1. Awareness of metrology in the academic dependability research community

In [Bondavalli 07a], the authors catalogue tools and experiments on the basis of the characteristics of the systems on which the tools were executed or the experiments were performed. The identified classification is the following: real-time/non real-time systems, centralized/distributed systems, safety critical/non-safety critical systems.

Before introducing the results of such classification, let us we briefly introduce the main metrological properties considered there (Uncertainty, Intrusiveness, Resolution, Repeatability)

Uncertainty provides quantitative information on the dispersion of the quantity values that could be reasonably attributed to the measurand and it has to be included as part of the measurement result. [BIMP 08] contains a comprehensive report of the methods of evaluating and expressing uncertainty in measurements.

It is well known that any measurement system perturbs the measurand, determining a modification of its value. Minimizing such perturbation, that is minimizing the system's **intrusiveness**, is therefore desirable when designing a measurement system.

Resolution is the ability of a measuring system to resolve among different states of a measurand. It is the smallest variation of the measurand that can be appreciated, i.e., that determines a perceptible variation of the instrument's output.

Repeatability is the property of a measuring system to provide closely similar indications in the short period, for replicated measurements performed i) independently on the same measurand through the same measurement procedure, ii) by the same operator, and iii) in the same place and environmental conditions.

Table 6.1 describes the *importance* of several metrological properties for the different categories of systems introduced above. Accordingly, it also describes the most important metrological properties that should be considered when measurement tools that shall provide reliable results are designed. A rank of the importance of assessing metrological properties in each configuration is provided by classifying properties as "*Recommended*" or "*Mandatory*".

Looking at Table 6.1, *intrusiveness* is considered a parameter of fundamental importance in the evaluation of computing systems, in particular for real-time systems: a tool being able to collect sufficiently reliable data in a non real-time environment, may collect unreliable data in a hard real-time environment. Intrusiveness is thus particularly critical in hard real-time systems, where timing predictability may be altered by the additional overhead of monitoring

Table 6.2 – Relevance of metrological properties relevance in measuring different kinds of systems

System Typology			Uncertainty	Intrusiveness	Resolution	Repeatability
Centralized	Non R-T	Non Critical	Recomm.	Recomm.	Recomm.	
Centralized	Non R-T	Critical	Recomm.	Recomm.	Recomm.	Recomm.
Centralized	Real-Time	Non Critical	Mandatory	Mandatory	Recomm.	
Centralized	Real-Time	Critical	Mandatory	Mandatory	Mandatory	Recomm.
Distributed	Non R-T	Non Critical	Recomm.	Recomm.		
Distributed	Non R-T	Critical	Recomm.	Recomm.		Recomm.
Distributed	Real-Time	Non Critical	Mandatory	Mandatory	Recomm.	
Distributed	Real-Time	Critical	Mandatory	Mandatory	Recomm.	Recomm.

tasks, or other mechanisms, e.g., fault injection probes.

Intrusiveness and *uncertainty* are related to each other since intrusiveness has consequences on uncertainty. This explains why in Table 6.1 all the rows in which intrusiveness is mandatory exhibit the same importance for uncertainty.

Resolution may be critical in real-time systems since it needs to be much lower than the imposed time deadline to allow useful quantitative evaluations of time or dependability metrics. In computing systems it can be generally assumed that resolution of the measurement system for time interval evaluation is equal to the granularity of the clock used in the experiment. In a centralized context it can happen that resolution is of the same order of magnitude of the measure, and it is thus of great importance to evaluate the resolution. On the other hand, when experiments are performed on distributed systems, uncertainty is usually far greater than resolution; in such cases, the evaluation and the control of resolution may be less crucial.

Repeatability is often not achievable when measurements are carried out on computer systems. In fact it can be hard to guarantee the same environmental conditions, especially in distributed systems, where differences among local clocks, in addition to the problems of thread scheduling and timing of events, enormously increase the difficulty of designing repeatable experiments.

The difficulty of reaching a satisfactory level for repeatability has been taken into account in some of the surveyed papers, but the ability to design repeatable experiments is generally occluded by limits on accurate time stamping.

Conclusive remarks from the work surveyed in [Bondavalli 07a] are that in general a full awareness of all metrological properties is lacking, and an exhaustive analysis of measurement parameters is performed only in a few cases. There is a wide-spread consciousness of the importance of intrusiveness, but there are few efforts that try to evaluate it with a rigorous approach. Regarding resolution, it is frequently not considered, even if it is usually the easiest parameter to estimate: the reason is probably that it is often considered not important, at least if compared with intrusiveness and uncertainty.

Moreover, the approach followed to quantitatively assess algorithms and systems is not univocal, but generally varies from a paper to another, making the comparison among different results quite difficult, if not meaningless (in the surveyed tools and experiments, results are never dealt with in terms of compatibility).

6.2. Resilience measurement, assessment and benchmarking in the field of academic metrology research

A survey of the most important and representative world congresses and international journals in the metrology field shows that the concept of dependability and resilience are not commonly addressed in the metrology research community.

Some years ago, in the most important conference of the field, IEEE Instrumentation and Measurement Technology Conference (IMTC), a paper was presented, which analyzed some possible failures of a distributed measurement or control architecture [Barger 03]. The paper was still preliminary, as it only took into consideration reliability of the measurement system, and seems to have had no continuation. In 2005, the editor of a technical journal, the IEEE Instrumentation & Measurement Magazine, wrote a short article, entitled "Dependability", introducing some basic dependability concepts to the instrumentation and measurement community [Fowler 05]. He focused on dependable (electronic) system design, and stressed the role of testing. The fact that such a "general" article is so recent and practically has had no evolution gives an idea of how little is the penetration of dependability concepts in the metrology research community.

Even less spread is the concept of *resilience*. In the very few cases in which the word resilience is mentioned, it either refers to a very specific concept [Hashempour 05] or it is not well defined [Tunstall 02, Kumar 04].

In particular, the work in [Hashempour 05] deals with the *error resilience* of some compressed codes used for telecommunication device testing. Error resilience is the capability of a compressed test data stream, which is transferred from automatic test equipment to the device under test, to tolerate errors. The effects of errors, such as bit flips, on the test data are analyzed and error resilience is evaluated.

In paper [Tunstall 02], the authors analyzed the resilience of hard disks to vibrations. In this case, resilience of the hard disk was not clearly defined, but they simply checked if vibrations prevent the hard disk from transferring data, that is hard disk resilience = ability to complete data transfer. What is interesting, the authors pay much attention to be as less intrusive as possible when experimentally measuring resilience.

Also in [Kumar 04], the authors present a theoretical model to calculate the performance/resilience of optimized multipath routing in variable network topology. They

deal with resilience of routing algorithms, but again do not clearly define what they mean by resilience.

6.3. Recent advances and open research problems

The previous analysis focused mainly on *time-related* dependability and resilience attributes. While time appears the most critical factor, many algorithms may have additional continuous attributes other than time that may suffer from the unawareness of the metrological properties previously described (e.g., location algorithms, which suffer from space uncertainty).

In [Bondavalli 07a] consciousness about metrological issues in the dependability research investigated. The results show that some general awareness about some metrological issues is indeed present, but the followed approaches are quite intuitive, and usually quite incomplete, as well. In particular, there is a lack of (i) a common systematic approach, (ii) wide-spread studies on feasibility of repeatable experiments, and (iii) diffusion of comparable results.

A preliminary result in the field of designing tools for measuring dependability properties of distributed systems is [Falai 07], where the author defines a conceptual framework for experimental evaluation and monitoring activities that supports a rigorous (from a metrological point of view) observation of distributed systems.

Very recently, the paper [Bondavalli *et al.* 2009b] has been accepted for publication in the most relevant journal of the Instrumentation and Measurement community. The paper presents a tool for dependability measurements in distributed systems that is capable of evaluating the uncertainty of measurement results based on distributed time measurements, and consequently discarding the results characterized by the largest uncertainty. After synthesizing the results of the survey presented in detail in [Bondavalli 07a], which has constituted the starting point for the development of the new tool, the paper describes the tool and includes the results of two case studies.

7. Fault Injection

A critical issue in the development of a resilient computer system is the validation of its fault-handling mechanisms[4]. Ineffective or unintended operation of these mechanisms can significantly impair the dependability of a computer system. Assessing the effectiveness and verifying the correctness of fault-handling mechanisms in computer systems is therefore of vital importance.

Fault injection is an important experimental technique for assessment and verification of fault-handling mechanisms. It allows researchers and system designers to study how computer systems react and behave in the presence of faults. Fault injection is used in many contexts and can serve different purposes, such as:

- Assess the effectiveness, i.e., fault coverage, of software and hardware implemented fault-handling mechanisms.
- Study error propagation and error latency in order to guide the design of fault-handling mechanisms.
- Test the correctness of fault-handling mechanisms.
- Measure the time it takes for a system to detect or to recover from errors.
- Test the correctness of fault-handling protocols in distributed systems.
- Verify failure mode assumptions for components or subsystems.

Over the years, many researchers have addressed the problem of validating fault-handling mechanisms by fault injection. Numerous papers on assessment and verification of fault-tolerant systems or individual mechanisms, and on fault injection tools have been published. This chapter gives an overview of the current state-of-the-art and selected important historical achievements in the area of fault injection.

Fault injection can in principle be carried out in two ways: faults can be injected either in *a real system* or in *a model of a system*. By a real system we mean a physical computer system, either a prototype or a commercial product. System models for fault injection experiments can be built using two basic techniques: *software simulation* and *hardware emulation*.

[4] We use the term fault-handling mechanism as an umbrella concept encompassing all mechanisms that handle faults and errors in a computer system. This includes mechanisms for achieving fault tolerance, preserving data integrity, ensuring safety, etc.

Hence, there are three main approaches to fault injection:

- *Injection of faults into real systems;*
- *Software simulation-based fault injection*, and
- *Hardware emulation-based fault injection*.

There are several techniques that can be used for injection of faults into real systems. These can be divided into three main categories: *hardware-implemented fault injection*, *software-implemented fault injection*, and *radiation-based fault injection*.

Software simulation-based fault injection can be performed in simulators operating at different levels of abstraction, such as device level, gate level, functional block level, instruction set architecture (ISA) level, and system level.

Hardware-emulation based fault injection uses models of hardware circuits implemented in large Field Programmable Gate Array (FPGA) circuits. These models can provide a highly detailed, almost perfect hardware representation of the system that is being verified or assessed.

Before we discuss and compare different fault injection techniques in detail, we must first introduce some terms and basic concepts. We use *target system* as a generic term for the system under assessment or verification. The target system executes a *workload*, which is determined by the program executed by the target system and the data processed by the program. The faults injected during the experiments constitute the *faultload*.

We distinguish between a *fault injection experiment* and a *fault injection campaign*. A fault injection experiment corresponds to injecting one fault and observing, or recording, how the target system behaves in presence of that fault. To gain statistical confidence in the assessment or the verification of a target system, we need to collect data from many fault injection experiments. A series of fault injection experiments conducted on a target system is called a fault injection campaign.

Fault injection techniques can be compared and characterized on the basis of several different properties. The following properties are applicable to all types of fault injection techniques:

- *Controllability* – ability to control the injection of faults in time and space.
- *Observability* – ability to observe and record the effects of an injected fault.

- *Repeatability* – ability to repeat a fault injection experiment and obtain the same result.
- *Reproducibility* – ability to reproduce the results of a fault injection campaign.
- *Reachability* – ability to reach possible fault locations inside an integrated circuit, or within a program.
- *Fault representativeness* – how accurately the faultload represents real faults.
- *Workload representativeness* – how accurately the workload represents real system usage.
- *System representativeness* – how accurately the target system represents the real system

The main advantage of performing fault injection in a real system is that the actual implementation of the fault-handling mechanisms is assessed and verified. Thus, system representativeness is usually higher when using a physical system compared to using software simulation or hardware emulation. On the other hand, fault models used in simulation-based and emulation-based fault injection can usually imitate real faults more accurately than artificial faults injected into a real system. Fault representativeness is therefore often higher for simulation-based and emulation-based fault injection. Also, controllability, observability, repeatability, reproducibility and reachability are normally higher in simulation-based and emulation-based fault injection compared to fault injection in real systems.

However, there are also several drawbacks and limitations to simulation-based and emulation-based fault injection. The development of the simulation/emulation model can increase development cost. Performing software simulations with an accurate simulation models can be very time-consuming. In fact, simulating a large amount of system activity (e.g., the execution of several million lines of source code) may not be feasible using a highly detailed model of the target system.

In simulation-based fault injection, it is therefore essential to make a trade-off between simulation time, on one hand, and the accuracy of the fault model(s) and the system model, on the other hand. Time overhead is much lower in hardware emulation-based fault injection than in software simulation-based fault injection. However, it may still be a concern in hardware emulation-based fault injection, e.g., in the verification of real-time systems.

All fault injection techniques have specific drawbacks and advantages. Several researchers have therefore proposed *hybrid fault injection* approaches, in which different techniques are combined in order to increase the scope and confidence in the verification or the assessment of a target system.

The remainder of this chapter is organised as follows. In the next section, we present tools and techniques for injection of hardware faults. Section 7.2 presents techniques for injection of software design and implementations faults. Section 7.3 presents fault injection techniques for assessment and testing of protocols used in fault-tolerant distributed systems. In Section 7.4, we list books and surveys of fault injection techniques. Finally, Section 7.5 concludes the chapter.

7.1. Techniques for injecting hardware faults

In this section, we describe techniques for injecting or emulating hardware faults. The first three subsections deal with *fault injection into real systems*, covering *hardware-implemented fault injection, software-implemented fault injection,* and *radiation-based fault injection*. Three additional subsections cover *software simulation-based fault injection*, *hardware emulation-based fault injection* and *hybrid fault injection*. Thus, hardware faults can be injected or emulated by all the techniques mentioned in the introduction of this chapter.

7.1.1. Hardware implemented fault injection

Hardware-implemented fault injection includes three techniques: *pin-level fault injection*, *power supply disturbances*, and *test port-based fault injection*.

In **pin-level fault injection**, faults are injected via probes connected to electrical contacts of integrated circuits or discrete hardware components. This method was used already in the 1950's for generating fault dictionaries for system diagnosis. Many experiments and studies using pin-level fault injection were carried out during the 1980's and early 1990's. Several pin-level fault injection tools were developed at that time, for example, MESSALINE [Arlat 90] and RIFLE [Madeira 94]. A key feature of these tools was that they supported fully automated fault injection campaigns. The increasing level of integration of electronic circuits has rendered the pin-level technique obsolete as a general method for evaluating fault-handling mechanisms in computer systems. The method is, however, still valid for assessment of systems where faults in electrical connectors pose major problem, such as automotive and industrial embedded systems.

Power supply disturbances (PSDs) are rarely used for fault injection because of low repeatability. They have been used mainly as a complement to other fault injection techniques in the assessment of error detection mechanisms for small microprocessors [Karlsson 91, Miremadi 95, Rajabzadeh 04]. The impact of PSDs is usually much more severe than the impact of other commonly used injection techniques, e.g., those that inject single bit-flips, since PSDs tend to affect many bits and thereby a larger part of the system state. Interestingly, some error detection mechanisms show lower fault coverage for PSDs than for single bit-flip errors [Rajabzadeh 04].

Test port-based fault injection encompasses techniques that use test ports to inject faults in microprocessors. Many modern microprocessors are equipped with built-in debugging and testing features, which can be accessed through special I/O-ports, known as test access ports (TAPs), or just test ports. Test ports are defined by standards such as the IEEE-ISTO 5001-2003 (Nexus) standard [*IEEE-ISTO 5001 03]* for real-time debugging, the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture (JTAG) [*IEEE Std 1149.1 01]*, and the Background Debug Mode (BDM) facility. Nexus and JTAG are standardized solutions used by several semiconductor manufacturers, while BDM is a proprietary solution for debugging developed by Freescale, Inc. Tools for test port-based fault injection are usually implemented on top of an existing commercial microprocessor debug tool, since such tools contain all functions and drivers that are needed to access a test port.

The type of faults that can be injected via a test port depends on the debugging and testing features supported by the target microprocessor. Normally, faults can be injected in all registers in the instruction set architecture (ISA) of the microprocessor. BDM and Nexus also allows injection of faults in main memory. Test ports could also be used to access hardware structures in the microarchitecture that are invisible to the programmer. However, information on how to access such hardware structures is usually not disclosed by manufacturers of microprocessors.

Tools that support test port-based fault injection include GOOFI [Aidemark 01] and INERTE [Yuste 03]. GOOFI supports both JTAG-based and Nexus-based fault injection, while INERTE is specifically designed for Nexus-based fault injection. An environment for BDM-based fault injection is described in [Rebaudengo 99].

Injecting a fault via a test port involves four major steps: i) setting a breakpoint via the test port and waiting for the program to reach the breakpoint, ii) reading the value of the target location (a register or memory word) via the test port, iii) manipulating this value and then

writing the new, faulty value back to the target location, and iv) resuming the program execution via a command sent to the test port.

The time overhead for injecting a fault depends on the speed of the test port. JTAG and BDM are low-speed ports, whereas Nexus ports can be of four different classes with different speeds. The simplest Nexus port (Class 1) is a JTAG port, which uses serial communication and therefore only needs 4 pins. Ports compliant with Nexus Class 2, 3 or 4 use separate input and output ports, know as auxiliary ports. These are parallel ports that use several pins for data transfer. The actual number of data pins is not fixed by the Nexus standard, but for class 3 and 4 ports the standard recommends 4 to 16 data pins for the auxiliary output port and 1 to 4 data pins for the auxiliary input port.

The main advantage of test port-based fault injection is that faults can be injected internally in microprocessors without making any alterations of the system's hardware or software. Compared to software-implemented fault injection, it provides better or equal capabilities of emulating real hardware faults. Finally, advanced Nexus ports (Class 3 and 4) provides outstanding possibilities for data collection and observing the impact of injected faults within a microprocessor. Existing tools have not fully exploited these possibilities. Hence, microprocessors with high-speed Nexus ports constitute interesting targets for the development of new fault injection tools, which potentially can achieve much better observability than existing tools do.

7.1.2. Software-implemented fault injection of hardware faults

Software-implemented fault injection encompasses techniques that inject faults through software executed on the target system.

There are basically two approaches that we can use to emulate hardware faults by software: *run-time injection* and *pre run-time injection*. In run-time injection, faults are injected while the target system executes a workload. This requires a mechanism that i) stops the execution of the workload, ii) invokes a fault injection routine, and iii) restarts the workload. Thus, run-time injection incurs a significant run-time overhead. In pre run-time injection, faults are introduced by manipulating either the source code or the binary image of the workload before it is loaded into memory. Pre run-time injection usually incurs less run-time overhead than run-time injection, but the total time for conducting a fault injection campaign is usually longer for pre run-time injection since it needs more time for preparing each fault injection experiment.

There are several fault injection tools that can emulate the effects of hardware faults by software, but they use different techniques for injecting faults and support different fault models. Most of these tools use run-time injection, since it provides better opportunities for emulating hardware faults than pre run-time injection.

Software-implemented fault injection relies on the assumption that the effects of real hardware faults can be emulated either by manipulating the state of the target system via run-time injection, or by modifying the target workload through pre run-time injection.

The validity of this approach varies depending of the fault type and where the fault occurs. Consider for example emulation of a *soft erro*r, i.e. a bit-flip error induced by a strike of a high energy particle. Flipping bits in main memory or processor registers can easily be done by software. On the other hand, the effect of a bit-flip in a processor's internal control logic can be difficult, if not impossible, to emulate accurately by software manipulations.

Emulating a permanent hardware fault requires a more elaborate set of manipulations than emulating a transient fault. For example, the emulation of a stuck-at fault in a memory word or a processor register would require a sequence of manipulations performed every time the designated word or register is read by a machine instruction. On the other hand, a transient fault requires only a single manipulation. The time overhead imposed by fault emulation thus varies for different fault types.

We here describe eight tools that are capable of emulating hardware faults through software. These tools represent important steps in the development of software-implemented fault injection for emulation of hardware faults. The tools are FIAT [Barton 90], FERRARI [Kanawati 92], FINE [Kao 93], DEFINE [Kao 95], FTAPE [Tsai 96], DOCTOR [Han 95], Xception [Carreira 98], MAFALDA [Arlat 02] and Exhaustif [Dasilva 07]. These tools use different approaches to emulating hardware faults and implement partly different fault models. Some of the tools also provide support for emulating software faults, as we describe in Section 7.2.

Researchers started to investigate software-implemented fault injection in the late 1980's. In the beginning, the focus was on developing techniques for emulating the effects of hardware faults. Work on emulation of software faults started a few years later.

One of the first tools that used software to emulate hardware faults was FIAT [Barton 90], developed at Carnegie Mellon University. FIAT injected faults by corrupting either the code or the data area of a program's memory image during run-time. Three fault types were

supported: *zero-a-byte*, *set-a-byte* and *two-bit compensation*. The last fault type involved complementing any 2 bits in a 32 bit word. Injection of single-bit errors was not considered, because the memory of the target system was protected by parity.

More advanced techniques for emulation of hardware faults were included in FERRARI [Kanawati 92], developed at the University of Texas, and in FINE [Kao 93], developed at the University of Illinois. Both these tools supported emulation of transient and permanent hardware faults in systems based on SPARC processors from Sun Microsystems. FERRARI could emulate three types of faults: *address line*, *data line*, and *condition code* faults, while FINE emulated faults in *main memory*, *CPU-registers* and the *memory bus*. DEFINE [Kao 95], which was an extension of FINE, supported fault injection in distributed systems and introduced two new fault models for intermittent faults and communication faults.

DOCTOR [Han 95] is a fault injection tool developed at the University of Michigan targeting distributed real-time systems. It supports three fault types: *memory faults*, *CPU faults* and *communication faults*. The memory faults can affect a single-bit, two bits, one byte, and multiple bytes. The target bit(s)/byte(s) can be set, reset and toggled. The CPU faults emulate faults in processor registers, the op-code decoding unit, and the arithmetic logic unit. The communication faults can cause messages to be lost, altered, duplicated or delayed. DOCTOR can inject transient, intermittent and permanent faults, and uses run-time injection for the transient and intermittent faults. Permanent faults are emulated using pre run-time injection.

FTAPE [Tsai 96] is a fault injector aimed at benchmarking of fault tolerant commercial systems. It was used to assess and test several prototypes of fault tolerant computers for on-line transaction processing. FTAPE emulates the effects of hardware faults in the CPU, main memory and I/O units. The CPU faults include single and multiple bit-flips and zero/set registers in CPU registers. The memory faults include single and multiple bit and zero/set faults in main memory. The I/O faults include SCSI and disk faults. FTAPE was developed at the University of Illinois in cooperation with Tandem Computers.

Xception [Carreira 98] is a fault injection tool developed at the University of Coimbra. This tool uses the debugging and performance monitoring features available in advanced microprocessors to inject faults. Thus it injects faults in a way which is similar to test port-based fault injection. The difference is that Xception controls the setting of breakpoints and performs the fault injections via software executed on the target processor rather than sending commands to a test port.

Xception injects faults through exception handlers executing in kernel mode, which can be triggered by the following events: *op-code fetch* form a specified address, *operand load* from a specified address, *operand store* to a specified address, and *a specified time* passed since start-up. These triggers can be used to inject both permanent and transient faults. Xception can emulate hardware faults in various functional units of the processors such as the integer unit, floating point unit and the address bus. It can also emulate memory faults, including *stuck-at-zero*, *stuck-at-one* and *bit-flip* faults. Xception is unique because it is one of very few academic tools that has been commercialised. Xception is sold by Critical Software, Coimbra, which released the first commercial version of the tool in 1999.

MAFALDA [Arlat 02] is a tool for assessment of commercial off-the-shelf microkernels. It uses software implemented fault injection to inject single or multiple bit-flips in the code and data segments of the microkernel under assessment. In addition, MAFALDA also allows corruption of input parameters during invocation of kernel system calls, and thus supports robustness testing of microkernels.

A more recent commercial tool, similar in functionality to Xception, is called Exhaustif [Dasilva 07]. It instruments the workload with a software component that injects faults at run-time. This component is configured through a communication interface (e.g., serial port, Ethernet) using a graphical user interface. It supports several fault models based on corruption of processor registers and memory, and interception of function calls. The software component that injects faults on the target system requires several kilobytes for code and data, which may be significant in terms of intrusiveness.

7.1.3. Radiation-based fault injection

Modern electronic integrated circuits and systems are sensitive to various forms of external disturbances such electromagnetic inference and particle radiation. One way of validating a fault tolerant system is thus to expose the system to such disturbances.

Although computer systems often are used in environments where they can be subjected to electromagnetic interference (EMI), it is not common to use such disturbances to validate fault tolerance mechanisms. The main reason for this is that EMI injections are difficult to control and repeat. In [Arlat 03], EMI was used along with three other fault injection techniques to evaluate error detection mechanisms in a computer node in a distributed real time system. A primary goal of this study was to compare the impact of pin-level fault injection, EMI, heavy-ion radiation and software-implemented fault injection. The study

showed that the EMI injections tended to “favour” one particular error detection mechanisms. For some of the fault injection campaigns almost all faults were detected by one specific CPU-implemented error detection mechanism, namely spurious interrupt detection. This illustrates the difficulty in using EMI as a fault injection method. Another attempt to use EMI for fault injection is reported in [Vargas 05].

A growing reliability concern for computer systems is the increasing susceptibility of integrated circuits to soft errors, i.e., bit-flips caused when highly ionizing particles hits sensitive regions within in a circuit. Soft errors have been a concern for electronics used in space applications since the 1970’s. In space, soft errors are caused by cosmic rays, i.e., highly energetic heavy-ion particles. Heavy-ions are not a direct threat to electronics at ground-level and airplane flight altitudes, because they are absorbed when they interact with Earth’s atmosphere. However, recent circuit technology generations have become increasingly sensitive to high energy neutrons, which are generated in the upper atmosphere when cosmic rays interact with the atmospheric gases. Such high energy neutrons are a major source of soft errors in ground-based and aviation applications using modern integrated circuits. All modern microprocessors manufactured in technologies with feature sizes below 90 nm are therefore equipped with fault tolerance mechanisms to cope with soft errors.

To assess the efficiency of such fault tolerance mechanisms, semiconductor manufacturers are now regularly testing their circuits by exposing them to ionising particles. The neutron beam facility at the Los Alamos Neutron Science Center (LANSCE) in the United States is often used for such tests as its energy spectrum is very similar to that of natural neutrons at sea-level. Similar neutron beam facilities are Osaka University’s RCNP and the ANITA Neutron Source at The Svedberg Laboratory in Uppsala, Sweden. Results of neutron beam testing are reported for Intel’s Itanium microprocessor in [Constantinescu05a], for the SPARC64 V microprocessor in [Ando 08], and for several generations of microprocessors from Sun Microsystems in [Dixit 09]. Sometimes proton radiation is used as a slightly less expensive alternative to neutron beam testing. Results of proton beam testing of the IBM POWER 6 processor can be found in [Kellington 07].

The sensitivity of integrated circuits to heavy-ion radiation can be exploited for assessing the efficiency of fault-handling mechanisms. In [Gunneflo 89] and [Karlsson 91], results from fault injection experiments conducted by exposing circuits to heavy-ion radiation from a Californium-252 source is reported. This method was also used in the previously mentioned study [Arlat 03], in which the impact of four different fault injection techniques was

compared. In this study, the main processor as well as the communication processor of a node in distributed system was exposed to heavy-ion radiation. The results showed that the impact of the soft errors injected by the heavy-ions varied extensively and that they activated many different error detection mechanisms in the target system.

Finally, we note that radiation-based fault injection has very low, or non-existent, repeatability. Due to low controllability, it is not possible to precisely synchronize the activity of the target system with the time and the location of an injection in radiation-based fault injection. Thus it is not possible to repeat an individual experiment. However, the ability to statistically reproduce results over many fault injection campaigns is usually high in particle radiation experiments. Both repeatability and reproducibility are low for EMI-based fault injection.

7.1.4. Simulation-based fault injection

As mentioned in the introduction, simulation-based fault injection can be performed at different levels of abstraction, such as the device level, logical level, functional block level, instruction set architecture (ISA) level, and system level. Simulation models at different abstractions layers are often combined in so called mix-mode simulations to overcome limitations imposed by the time overhead incurred by detailed simulations.

FOCUS [Choi 92] is an example of a simulation environment that combines device-level and gate-level simulation for fault sensitivity analysis of circuit designs with respect to soft errors.

At the logic level and the functional block level, circuits are usually described in a hardware description language (HDL) such as Verilog or Very High Speed Integrated Circuit Hardware Description Language (VHDL). Several tools have been developed that support automated fault injection experiments with HDL models, e.g., MEFISTO [Jenn 94] and the tool described in [Delong 96]. There are several different methods for implementing the fault injection process, such as modifying the HDL code [Assaf 04], modifying the HDL simulator, commanding the simulator through scripts or, in a more recent example [Das 06], using the *force* and *release* constructs in Verilog to emulate stuck-at faults.

Recently, several studies aimed at assessing the soft error vulnerability of complex high-performance processors have been conducted using simulation-based fault injection. In [Wang 06] a novel low-cost approach for tolerating soft errors in the execution core of a high-performance processor is evaluated by combining simulations in a detailed Verilog model

with an ISA-level simulator. This approach allowed the authors to study the impact soft errors for seven SPEC2000 integer benchmarks through simulation.

DEPEND [Goswami 97] is a tool for simulation-based fault injection at the functional level aimed at evaluating architectures of fault-tolerant computers. A simulation model in DEPEND consists of number of interconnected modules, or components, such as CPUs, communication channels, disks, software systems, and memory. DEPEND is intended for validating system architectures in early design phases and serves a complement to probabilistic modelling techniques such as Markov and Petri net models. DEPEND provides the user with predefined components and fault models, but also allows the user to create new components and new fault models, e.g., the user can use any probability distribution for the time to failure for a component.

7.1.5. Hardware emulation-based fault injection

The advent of large Field Programmable Gate Arrays (FPGAs) circuits has provided new opportunities for conducting model-based fault injection with hardware circuits. Circuits designed in a HDL are usually tested and verified using software simulation. Even if a powerful computer is used in such simulations, it may take considerable time to verify and test a complex circuit adequately. To speed up the test and verification process, techniques have been developed where HDL-designs are tested by hardware emulation in a large FPGA circuit. This technique also provides excellent opportunities for conducting fault injection experiments. Hardware emulation-based fault injection has all the advantages of simulation-based fault injection such as high controllability and high repeatability, but requires less time for conducting a fault injection experiment compared to using software simulation.

The use of hardware emulation for studying the impact of faults was first proposed in [Kwang-Ting 99]. The authors of that paper used the method for fault simulation, i.e., for assessing the fault coverage of test patterns used in production testing.

Fault injection can be performed in hardware emulation models through *compile time reconfiguration* and *run-time reconfiguration*. Here reconfiguration refers to the process of adding hardware structures to the model which are necessary to perform the experiments. In compile-time reconfiguration, these hardware structures are added by instrumentation of the HDL models. An approach for compile-time instrumentation for injection of single event upsets (soft errors) is described in [Civera 03]. This work presents different instrumentation

techniques that allow injection of transient faults in sequential memory element as well as in microprocessor-based systems.

One disadvantage of compile-time reconfiguration is that the circuit must be re-synthesised for each reconfiguration, which can impose a severe overhead on the time it takes to conduct a fault injection campaign. In order to avoid re-synthesizing the target circuit, a technique for run-time reconfiguration is proposed in [Antoni 03]. This technique relies on directly modifying the bit-stream that is used to program the FPGA-circuit. By exploiting partial reconfiguration capabilities available in some FPGA circuits, this technique achieved substantial time-savings compared to other emulation-based approaches to fault injection.

A tool for conducting hardware emulation-based fault injection called FADES is presented in [Andrés 06, Andrés 08]. This tool uses run-time configuration and can inject several different types of transient faults, including bit-flips, pulse, and delay faults, as wells as faults that cause digital signals to assume voltage levels between "1" and "0".

Even though FPGA-based techniques overcome the performance issues present in software simulation, it is often difficult to obtain ideal observability due to the communication required for observing the behaviour of emulated circuits. A recently proposed method [Ejlali 07] reduces this overhead by having a single combinational circuit for both the faulty circuit and the fault-free circuit. The complete circuit repeatedly executes one clock cycle with the fault-free flip-flops followed by one clock cycle with the faulty flip-flops, and the output is multiplexed to a comparator, in order to identify which faults cause errors on the target. This method provides good observability under the bit-flip fault model and avoids duplicating the entire combinational circuit, which is assumed to be unaffected by faults.

There is a concern with representativeness when using hardware emulated circuits, rather than the actual hardware. In [Ramachandran 08], the results of fault injection on a hardware-emulated IBM POWER6 processor (the authors call it "hardware accelerated simulation") are compared to radiation-based fault injection. The results show a close match between the two techniques, therefore providing evidence in favour of hardware emulation-based fault injection.

7.1.6. Hybrid approaches for injecting hardware faults

Hybrid approaches to fault injection combine several fault injection techniques to improve the accuracy and scope of the verification, or the assessment, of a target system.

An approach for combining software-implemented emulation of hardware faults and simulation-based fault injection is presented in [Guthoff 95]. In this approach, the physical

target is run until the program execution hits a fault injection trigger, which causes the physical system to halt. The architected state of the physical system is then transferred to the simulation model, in which a fault is injected, e.g., in the non-visible parts of the microarchitecture. The simulator is run until the effects of the fault have stabilized in the architected state of the simulated processor. This state is then transferred back to the physical system, which subsequently is restarted so that the system-level effects of the fault can be determined.

An extension of the FERRARI tool which allows it to control a hardware fault injector is described in [Kanawati 95]. The hardware fault injector can inject logic-0/logic-1 faults into the memory bus lines of a SPARC 1 based workstation. The authors used the hardware fault injector to study the sensitivity of the computer in different operational modes. The results showed that system was more likely to crash from bus faults when the processor operated in kernel mode, compared to when it operated in user mode. This study showed that it is feasible to extend a tool for software-implemented fault injection with other techniques at reasonable cost, since many of the central functions of a tool are independent of the injection technique.

A more recent tool that supports the use of different fault injections techniques is NFTAPE [Stott 00], developed at the University of Illinois. This tool is aimed at injecting faults in distributed systems using a technique called LightWeight Fault Injectors (LWFI). The purpose of the LWFI is to separate the implementation of the fault injector from the rest of the tool. NFTAPE provides a standardized interface for the LWFIs, which simplifies the integration and use of different types of fault injectors. NFTAPE has been used with several types of fault injectors using hardware-implemented, software-implemented, and simulation-based fault injection.

7.2. Techniques for injecting or emulating software faults

Software faults are currently the dominating source of computer system failures. Making computer systems resilient to software faults is therefore highly desirable in many application domains. Much effort has been invested by both academia and industry in the development of techniques that can tolerate and handle software faults. In this context, fault injection plays an important role in assessing the efficiency of these techniques. Hence, several attempts have been made to develop fault injection techniques that can accurately imitate the impact of real software faults.

The current state-of-the-art techniques in this area rely exclusively on software-implemented fault injection. There are two fundamental approaches to injecting software faults into a computer system: *fault injection* and *error injection* [Durães 06]. Fault injection imitates mistakes of programmers by changing the code executed by the target system, while error injection attempts to emulate the consequences of software faults by manipulating the state of the target system.

Regardless of the injection technique, the main challenge is to find fault sets or error sets that are representative of real software faults. Other important challenges include the development of methods that allow software faults to be injected without access to the source code, and techniques for reducing the time it takes to perform an injection campaign. First, we discuss emulation of software faults by error injection, and then software fault injection.

7.2.1. Emulating software faults by error injection

There are two common techniques for emulating software faults by error injection: *program state manipulation* and *parameter corruption*. Program state manipulation involves changing variables, pointers and other data stored in main memory or CPU-registers. Parameter corruption corresponds to modifying parameters of functions, procedures and system calls. The latter is also known as API parameter corruption and falls under category of robustness testing, which is discussed in Chapter 13. Here we discuss techniques for emulating software faults by program state manipulation.

Many of the tools that we described in conjunction with emulation of hardware faults through software-implemented fault injection, e.g., FIAT [Barton 90], FERRARI [Kanawati 92], FTAPE [Tsai 96], DOCTOR [Han 95],Xception [Carreira 98], and MAFALDA [Arlat 02] can potentially be used to emulate software faults since they are designed to manipulate the system state. However, none of these tools provide explicit support for defining errors that can emulate software faults and the representativeness of the injected faults is therefore questionable.

An approach for generating representative error sets that emulates real software faults is presented in [Christmansson 96]. This approach was based on a study of software faults encountered in one release of a large IBM operating systems product. Based on their knowledge of the observed faults, the authors developed a procedure for generating a representative error set for error injection. The study addressed four important questions related to emulation of software fault by error injection: *what* error model(s) should be used;

where should errors be injected; *when* should errors be injected; and *how should a representative operational profile* (workload) be designed? This work shows the feasibility of generating representative error sets when data on software faults is available.

An experimental comparison between fault and error injection is presented in [Christmansson 98]. Fault and error injection experiments were carried on a safety-critical real-time control application. A total of 200 *assignment*, *checking* and *interface* faults were injected by mutating the source code, which was written in C. The failure symptoms produced by these faults were compared with failure symptoms produced by bit flip errors injected in processor registers, and in the data and stack areas of the main memory. A total 675 errors were injected. A comparison of the failure distributions were made for eight different workload activations (test cases).

The authors conclude that the choice of test case caused greater variations in the distribution of the failure symptoms than the choice of fault type, when fault injection was used. On the other hand, for error injection the choice of error type caused greater variations in the failure distribution than the choice of test case.

There were also significant differences between the failure distributions obtain with fault injection and with error injection. The authors claim that these differences occurred because a time-based trigger was used to control the error injections. They also claim that the fault types considered could be emulated more or less perfectly by using a break-point based trigger, although no experimental evidence is presented to support this claim. This study points out that it may be difficult to find error sets that emulate software faults accurately, and that the selection of the test case (workload activation) is as important as the selection of the fault/error model for the outcome of an injection campaign.

7.2.2. Techniques for injection of software faults

An obvious way to inject software faults into a system is to manipulate the source code, object code or machine code. Such manipulations are known as *mutations*. Mutations have been used extensively in the area of program testing as a method for evaluating the effectiveness of different test methods. They have also been used for the assessment of fault-handling mechanisms.

The studies presented in [Ng 96] and [Ng 99] inject software faults in an operating system through simple mutation of the object code. The primary goal of the fault models used in

these studies was to generate a wide variety of operating system crashes, rather than achieving a high degree of representativeness with respect to real soft faults.

FINE [Kao 93] and DEFINE [Kao 95] were among the first tools that supported emulation of software faults by mutations. The mutation technique used by these tools requires access to assembly language listings of the target program. FINE and DEFINE emulates four types of software faults: *initializations*, *assignment*, *condition check*, and *function* faults. These fault models were defined based on experience collected from studies of field failure data.

An interesting technique, called Generic Software Fault Injection Technique (G-SWFIT), for emulation of software faults by mutations at the machine-code level is presented in [Durães 02]. This technique analyses the machine code to identify locations that corresponds to high-level language constructs that often results in design faults. The main advantage of G-SWFIT is that software faults can be emulated without access to the source code. A set of operators for injection of representative software faults using G-SWFIT was presented in [Durães 03]. These operators were derived from a field failure data study of more than 500 real software faults. These two works jointly represent a unique contribution, since they provide the first fault injection environment that can inject software faults which have been proven to be representative of real software faults. They also constitute the foundation of a methodology for definition of faultloads based on software faults for dependability benchmarking presented in [Durães 06].

7.3. Techniques for testing protocols for fault-tolerant distributed systems

Several fault injection tools and frameworks have been developed for testing of fault-handling protocols in distributed systems. The aim of this type of testing is to reveal design and implementation flaws in the tested protocol. The tests are performed by manipulating the content and/or the delivery of messages sent between nodes in the target system. We call this message-based fault injection. It resembles robustness testing discussed in Chapter 13 in the sense that the faults are injected into the inputs of the target system.

A careful definition of the failure mode assumptions is crucial in the design of distributed fault-handling protocols. The failure mode assumptions provide a model of how faults in different subsystems (computing nodes, communication interfaces, and networks) affect a distributed system. A failure mode thus describes the impact of subsystem failures in a distributed system. Commonly assumed failure modes include Byzantine failures, timing failures, omission failures, crash failures, fail-stop failures and fail-signal failures. At the

system-level, these subsystem failures correspond to faults. Hence, tools for message-based fault injection intend to inject faults that correspond to different subsystem failure modes.

The experimental environment for fault tolerance algorithms (EFA) [Echtle 92] is an early example of a fault injector for message-based fault injection. The EFA environment provides a *fault injection language* that the protocol tester uses to specify the test cases.

The tool inserts fault injectors in each node of the target system and can implement several different fault types, including message omissions, sending a message several times, generating spontaneous messages, changing the timing of messages, and corrupting the contents of messages. A similar environment is provided by the DOCTOR tool [Han 95], which can cause messages to be lost, altered, duplicated or delayed.

Specifying test cases is a key problem in testing of distributed fault-handling protocols. A technique for defining test cases from Petri-net models of protocols in the EFA environment is described in [Echtle 95]. An approach for defining test cases from an execution tree description of a protocol is described in [Avresky 96].

A framework for testing distributed applications and communication protocols called ORCHESTRA is described in [Dawson 95, Dawson 96]. This tool inserts a probe/fault injection layer (PFI) between any two consecutive layers in a protocol stack. The PFI layer can inject deterministic and randomly generated faults in both outgoing and incoming messages. ORCHESTRA was used in a comparative study of six commercial implementations of the TCP protocol reported in [Dawson 97].

Neko [Urban et al 2001] is another framework for testing distributed applications. It supports rapid prototyping of distributed algorithms and, with the NekoStat extension [Falai et al 05], it can be used to perform quantitative evaluations of distributed systems. Neko provides supports for fault injection, e.g. injection of faults between consecutive layers in the application stack. The core part of Neko is pure Java (J2SE) and so Neko processes are able to work on top of many different OSs and platforms. Neko provides support for both simulation and experimental execution of distributed applications.

The failure of a distributed protocol often depends on the global state of the distributed system. It is therefore desirable for a human tester to control the global state of the target system. This involves controlling the states of a number of individually executing nodes, which is a challenging problem. Two tools that address this problem are CESIUM [Alvarez 97] and LOKI [Chandra 00]. An environment for message-based fault injection for

assessment of OGSA middleware for grid computing is presented in [Looker 03]. A similar tool for testing of Web-services, called WS-FIT, is presented in [Looker 05a]. This fault injector can decode and inject meaningful faults into SOAP-messages. It uses an instrumented SOAP API that includes hooks allowing manipulation of both incoming and outgoing messages. A comparison of this method and fault injection by code insertion is presented in [Looker 05b]. Another tool aimed at testing grid services and cluster applications is Fail-FCI [Hoara 05]. This tool is based on FAIL, a fault injection language that is used to define failure scenarios in a distributed system using a state machine approach. The Fail-FCI tool is extended in [Hoara 06] to allow fault injection in distributed Java applications. Results from message-based fault injection assessment of several implementations of CORBA middleware are reported in [Marsden 02]

7.4. Surveys and books on fault injection

Surveys of fault injection tools and techniques can be found in [Clark 95, Hsueh 97]. Both these surveys are more than ten years old and are therefore partly outdated. An overview of tools for benchmarking and fault injection in grid computing is presented in [Tixeuil 06].

Two books that address fault injection are *Software fault injection: inoculating programs against errors* by Jeffrey Voas and Gary McGraw [Voas 98] and *Fault injection techniques and tools for embedded systems reliability evaluation* by Benso and Prinetto (Eds.) [Benso 03].

7.5. Concluding remarks

We have provided an overview of the state-of-the-art and historical achievements in the field of fault injection. The overview covers tool and techniques for injecting three main fault types: physical hardware faults, software design and implementation faults, and faults affecting messages in distributed systems. We have not included papers that discuss fault injection as a method to conduct security intrusions and cracking cryptographic ciphers, so called fault attacks. Overviews of such fault injection methods can be found in [Bar-El 06] and [Chong 07]. The topic of attack injection for evaluation of intrusion detection systems is covered in Chapter 10 and 12.

Our intention has been to highlight the most important fault injection tools and techniques used for assessment and test of dependable computer systems, but we do not claim to have covered all works in the field. This would not have been possible since the number of publications that deal with fault injection is quite large. A search on Google Scholar lists

about 5000 scientific publications that mention the exact phrase “fault injection” in the text of the document.

Our overview shows that many fault injection tools and techniques have been proposed since the late 1980’s. Over the last ten years we have seen a steady increase in the number of publications related to fault injection. Google Scholar lists 623 papers published in the years 2000 to 2002 that mention “fault injection” in the text. The corresponding figure for 2006 to 2008 is 877. (These figures was obtained June 20, 2009 and do not include publications on security attacks). As fault injection techniques and tools have matured, we see that recently developed tools tends to improve and refine existing methods rather than exploiting new principles for fault injection.

8. Field Measurements

Field measurements refer to observations of systems in the operational phase, i.e., systems that are actually in use. The results obtained from these observations have the very important characteristic of being realistic: the operation conditions and environment, and the workload are not mere experimental approximations. Very often, field measurements are not representative as there is no guarantee that all possible, important system configurations have been observed. Nevertheless, field measurements and field data are a unique and very important source of information for researchers when studying the dependability properties systems, such as availability, reliability and robustness.

Concerning data on the robustness of the computer-based systems, field data is mostly obtained from reports (bug reports, incident reports, security logs, and so on, depending on the nature of the incident). These reports are filed by the users and operators and are typically used by the system developers to solve the incidents and improve the system. Observations made in closed-source, proprietary systems are typically not available to the public. Observations originating from open-source systems are normally made available to the community (e.g., stored in a repository).

There are basically two driving forces behind the collection of field data: development and research. The first is committed to the improvement of specific systems and to solve problems on those specific systems that are discovered during the operational phase. The second driving force aims to understand the issues related to systems reliability and dependability and to propose new techniques to increase the reliability of non-specific (non vendor-specific) systems. A third driving force is a market-driven one, to promote awareness of a given product (e.g., network providers, such as sprint and AT&T, publish their performance and dependability data to promote the company and attract new customers). However, the first two driving forces are those more relevant to research works.

The research driving force, although not tied to specific vendors or industry goals, is necessarily dependent on the existence of data. This data is mainly that which was collected by users or operators and is not related to any research goal. Thus, so far, the main origin of field data is the occurrence of incidents. This fact has an overwhelming impact on the nature of the available data, which is mainly related to computer failures and security incidents.

The most complex and error prone components of computer-based systems are the software. Thus, software faults represent a large part of the relevant and available field data. Security issues are necessarily tied to the availability of the system to users. Networks are the mechanism that most exposes a system to the general public and represents the highest risk to security-related incidents. As web-based systems are currently the majority of network-enabled systems, most of the field data on security is related to web-based systems.

Although field data (field measurements) is highly relevant to the research community to understand and improve computer-based systems robustness and reliability, the availability of such data remains hard to obtain. The few data available is based on open-source projects and published research works. The importance of field data is widely recognised among researchers as shown in workshops such as *RAF07: Reliability Analysis of System Failure Data* organised by Microsoft Research in Cambridge and Darmstadt University in 2007. Each open-source development team or research team presents its own data and its own view. One important initiative to mitigate the scarcity and fragmented view of field data is the development of public repositories, to store data and results based on that data originating from many sources and teams.

8.1. Overview of field data on software faults

In [Gray 85] a survey on failures of Tandem fault-tolerant computer systems is presented. This study was based on the 166 failure reports over a period of seven months covering more than two thousand systems. From those 166 failures, 59 were caused directly by the novelty of the software or hardware of the systems under consideration (termed as infant mortality failures). It is worth noting that although the systems used in this work are considered as fault tolerant systems, and great efforts were put on the development to avoid bugs in the software, still there are defects in deployed systems. This suggests that the defects existing in the software are only activated in rare conditions that elude even a very thorough testing phase.

Another study by the same author [Gray 90] five years later provides an interesting opportunity to compare the trend evolution using the same type of systems. This work used a much larger database of field data composed of 515 failures covering a period of five years. An important difference compared to the previous work is the scenario of the infant mortality failures. In the work done five years before [Gray 85] most of this type of failure was related to hardware problems, while in [Gray 90] the majority of the causes of infant mortality failure is software related. The comparison of two field data sets allows researchers to draw an

important conclusion regarding faulty type trend evolution. In this case at hand software faults are increasingly becoming the root cause behind system failures. According to [Gray 90], software faults are responsible for 62% of the total failures (roughly double the figure from the 1985 work). Observing the data presented for individual years covered by the work, it becomes evident that the number of failures caused by operation, environment, maintenance and hardware decreases, particularly in hardware and maintenance cases, while the number of failures caused by software defects remains approximately stable. The lack of improvement in software is explained by the increased size and complexity of the software.

Another relevant study using field data is presented in [Lee 93] and [Lee 95]. This work analyzed two hundred failure reports (Tandem product report – TPR) related to the GUARDIAN90 operating system covering four months in 1991. Of those, at least 153 were caused by software problems. This amounts to 76%, a figure even greater than those presented in [Gray 85] and [Gray 90]. The amount of failures caused by software may be even larger as there are 13% of failures whose cause was not determined and may or may not be caused by software issues. This work also provides an overview on the operating system error detection: The major errors detected were: 48 address violations, 19 kernel consistence check errors, and 33 non-kernel consistency check errors. This view provides an insight on which errors are likely to be caused by software faults.

A study on field failures in Windows NT networked systems is presented in [Xu 99]. This work describes failures from downtime viewpoint. According to its findings, software related failures (counting both system and application software) is responsible for 23% of the total downtime. On the other hand, hardware related problems are responsible for only 10% of the total downtime. This work shows an important perspective on software faults: not only they are the dominant cause for field-observed failures in terms of number of occurrences, but also are responsible for most of the downtime. This work also identifies planned maintenance actions as responsible for 24% of all downtime. Considering that planning downtimes are not quite a fault (as they are planned), if we remove those 24% from the statistics and normalize the remaining figures, the downtime caused by software problems alone will jump to 30.2%.

A quantitative analysis concerning several properties of software faults is presented in [Fenton 00]. This work used data obtained from two releases of a commercial system providing a rare opportunity to observe the relationship between faults discovered during the development and faults discovered after deployment. The most interesting findings of this work are the following: a small number of modules contain most of the software faults of the

entire system; a small number of modules contain most of the faults that cause failures; and fault densities (number of fault discovered divided by a measure of module size, usually KLOC) at corresponding phases of testing of consecutive releases of the same software system remain roughly the same. The latter supports the idea that successive releases of a given software product do not necessarily means increased reliability and that hidden faults will always exist even in mature software.

The work presented in [Duraes 06] used software faults on several open-source projects to characterize the nature of the most common software faults and to develop a technique to accurately emulate them. That work was based on 668 faults discovered in the operational phase. An important conclusion from this work is the identification of the most common software faults that are likely to exist in deployed software systems. This was made possible only through the existence of large quantities of bug reports available to researchers (in this case, available at the sourceforge site [SourceForge 08].

8.2. Overview of field data on web-sever security incidents

Security in the cyber-world is a major concern that is receiving more and more attention from the research community. However, in spite of this growing awareness of security aspects at web application level, there is an increase in the number of reported attacks. A recent example (August 17, 2007) of such attacks occurred in the recruitment website Monster.com where 1.6 million of personal records were stolen [Vnunet 07].

According to a Computer Crime and Security Survey [Gordon 05] done by the FBI in 2005, around 45% of the respondents had reported unauthorized access to information estimating a loss of $31 233.100 and a loss of $30 933 000 due to theft of proprietary info. Up to 56% of the respondents reported unauthorized use of computer systems and 13% did not know if they have been attacked. Furthermore, 95% of the correspondents reported more than 10 web site incidents.

In a survey [Gordon 06] it is concluded that defacement of web sites is a problem of many organizations, as 92% of the respondents reported more than 10 web site incidents. According to Gartner, 75% of the attacks are on web based applications [Gartner 08]. According to an Acunetix audit result [Acunetix 07], 70% of the 3,200 websites scanned in the past three years contain security vulnerabilities. The NTA Monitor's 2007 Annual Security Report [NTA 07] states that online risks in financial institutions have increased 16%, and 28% in publishing

companies. Overall, 32% of the websites analyzed contain critical vulnerabilities that are widely known and actively exploited by hackers.

The Open Web Application Security Project (OWASP Foundation) released its ten most critical web application security vulnerabilities 2007 [Stock 07] based on data provided by Mitre vulnerability type distributions in Common Vulnerabilities Exposed (CVE) [Christey 07]. In this report XSS is ranked as the most critical vulnerability, followed by Injection Flaws, particularly SQL injection.

[Fonseca 08] presents a field study analyzing 679 security patches of six widely used web applications. Results are compared against other field studies on general software faults (i.e., faults not specifically related to security), showing that only a small subset of software fault types is related to security. Furthermore, the detailed analysis of the code of the patches has shown that web application vulnerabilities result from software bugs affecting a restricted collection of statements. A detailed analysis of the conditions/locations where each fault was observed in the field study is presented allowing future definition of realistic fault models that cause security vulnerabilities in web applications.

8.3. Overview of data repositories

Data repositories are an excellent resource to store and share information for research purposes. One type of valuable information that can be shared through data repositories is the results from field data studies. Although data repositories to store failure data and dependability experiments results are relatively rare (especially considering the huge value of real failure data to help designers in improving computer systems), several initiatives have been proposed and are currently available.

The Data & Analysis Center for Software (DACS) is a Department of the US Defense Information Center supporting research on software reliability and quality. It serves as centralized source for data related to software metrics. The DACS maintains the Software Life Cycle Experience Database (SLED). This repository is intended to support the improvement of the software development process. The SLED is organized into nine data sets covering all phases and aspects of the software lifecycle ([DACS 08] and [Delude 95]). Examples of these datasets are:

- The DACS Productivity Dataset (collected from government and private industry sources). This dataset consists on data on over 500 software projects and is mainly oriented to software cost modelling and productivity analysis [Nelson 78]. The data

represents software from early 60s to early 80s and includes software projects ranging from avionics to off-the-shelf packages. The information in this dataset includes the following: size of project, effort, language, schedule, errors.

- The NASA/SEL Dataset (contributed by the Software Engineering Laboratory (SEL) at NASA Goddard Space Flight Center). This repository maintains data on avionic applications since 1976. The dataset is available by request on disk and it can be accessed through web browser. Using the latter, users have access to analytical summaries including linear regression, scatter plots and histograms. The analytical results are created dynamically per request during the HTTP session and served to the user. The repository information is stored in a relational database and the link between the data repository and the web server is supported through Perl applications.
- The Software Reliability Dataset (collected at Bell Laboratories) [Musa 80]. This repository describes failures in a wide range of application domains including real time, control, office, and military applications. This dataset was primarily aimed at the validation of software reliability models and to assist software managers to monitor and predict software tests. As in the NASA/SEL dataset, the information can be obtained by request, and it can also be accessed through web interface.

The Metrics Data Program (MDP) Repository is a database maintained by the NASA Independent Verification and Validation facility [NASA 08]. The repository is aimed at the dissemination of non-specific data to the software community and it is made available to the general public at no cost. All the data available in the repository is sanitized by the projects representatives, and all the necessary clearances are provided. Users of the repository are free to analyze the data for their specific research goals.

The MDP repository is part of the MDP on going effort to improve the ability to predict error in software by improving the quality of the problem data related to software (e.g., improve the quality of the information about the relationship of the error and the development phase). To this effort, the MDP recruits the participations of private-sector and public-sector projects. Recruited projects maintain complete control of data release and the level of participation in the program. The effort required by the participating projects is minimal.

The repository contains data on the software projects that were collected and validated by the MDP program, spanning more than 8 years and including more than 2700 error reports. The information stored in the repository consists of error data and software metrics and error data

at the function/method level. The dataset enables data associations between products and metrics, and problems with the Orthogonal Defect Classification (ODC) [Chillarege 95].

The Software Reference Fault and Failure Data Project [NIST 08] is maintained by the National Institute of Standards & Technology and is aimed at the development of metrology, taxonomy and repository for reference data for software assurance. The project maintains a repository on software fault data specifically aimed at helping industry protect against releasing software systems with faults and to help assess software systems quality by providing statistical methods and tools. The repository is available to the public upon request. The access to the information online allows users to view data and execute simple queries. Analytical and statistical use of the data is possible through a program developed within the project and available to the public (the EFFTool).

The Computer Failure Data Repository (CFDR) is a public repository on computer failure data ([Schroeder 07a] and [CFDR 08]) supported by USENIX. The repository is aimed at the acceleration of the research on system reliability with the ultimate goal of reducing or avoiding downtime in computer systems. To this goal, the CDFR hopes to remove the main difficulty faced by researchers, which is the lack of reliable and precise information about computer failures.

The CDFR repository is open to both obtaining and contributing data. The repository comprises nine independent data-sets focusing mainly on very large storage systems. The repository information covers many aspects, including: software failures, hardware failures, operator errors, network failures, and operational environment problems. The raw data is available to the public [CFDR 08] through web interface. The project does not offer online capability for analytic and statistical data-processing.

The usefulness of public repositories to the research communities is demonstrated by the existence of studies based on the information stored in publicly available repositories (e.g. [Schroeder 07b]). Nevertheless, and in spite of the different repository initiatives already available, the raw data from the vast majority of research works on experimental dependability evaluation and on field failure data is not available in any repository. Hundreds of papers have been published but the raw data that have led to the final results presented in those papers is not available.

8.4. Historical perspective

Field measurements and field data studies have always been a major support for science and technical advancement. This is particularly true for disciplines strongly tied to experimental research. Thus, field measurements have consistently been recognized as a very important resource for computer robustness, dependability and security, among other areas. One seminal and influential study based on field measurements and field data on computer robustness was the study on failures of Tandem fault-tolerant computer systems in 1985 [Gray 85]. This study provided valuable information on the behaviour of systems in real operation scenarios and enabled researchers to assess the evaluation of robustness-related techniques by comparing these results with those of later and similar surveys (e.g., [Gray 90] or [Lee 93]). The relevance of field measurements and field data studies to the academic community can be gauged by the longevity of the publications: [Gray 85], although published more than 20 years ago, continues to be cited in recent works. More recent works also show this community receptivity which confirms the need for such works. In fact, one could argue that the main reason for not having more such works published is the fact that they usually require a large effort from the researcher, particularly of time. The apparent scarcity of field measurements and field data studies makes the existence of coordinated efforts to share data (e.g., through web-based databases) more relevant.

9. Modelling and Model-based Assessment

As defined in [Balbo 01], a model is an abstraction of a system "that highlights the important features of the system organization and provides ways of quantifying its properties neglecting all those details that are relevant for the actual implementation, but that are marginal for the objective of the study". There exist several types of models, and the choice of a proper model depends on many factors, like the complexity of the system, the specific aspects to be studied, the attributes to be evaluated, the accuracy required, the questions to be answered in the system, and the resources available for the study.

Models play a primary role in the resilience assessment process of modern computing systems.

First of all, they allow an "a posteriori" resiliency analysis, to understand and learn about specific aspects, to detect possible design weak points or bottlenecks, to perform a late validation of the dependability requirements and to suggest sound solutions for future releases or modifications of the systems. Furthermore, assessing the resilience of composite systems, often including a dynamic mixture of components built by different parties and for different purposes, is a difficult task that may require the combination of several assessment methods and approaches. In this perspective, models can be profitably used as support for experimentation and vice-versa. On one side, modelling can help in selecting the features and measures of interest to be evaluated experimentally, as well as the right inputs to be provided for experimentation. On the other side, the measures assessed experimentally can be used as parameters in the models, and the features identified during the experimentation may impact the semantic of the dependability model.

This chapter provides an overview of the state of knowledge related to stochastic model-based assessment approaches, which are most commonly used for resiliency evaluation of current computing systems. An overview of the history in model-based assessment research, as well as current directions, is sketched in Figure 9.1.

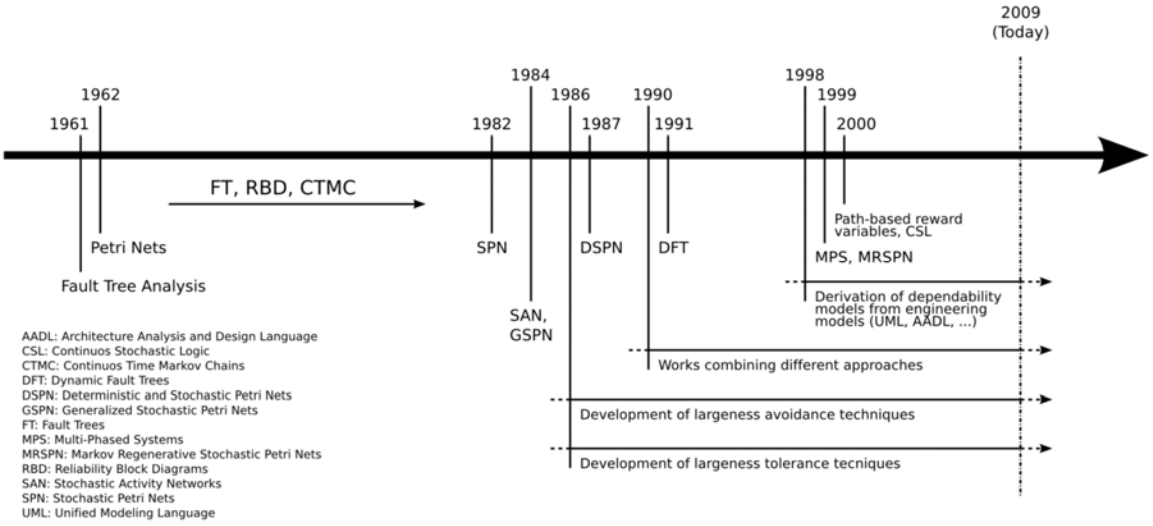


Figure 9.1: Historical timeline of advances in modeling and model-based assessment

The approaches presented in this figure will be explained and discussed in the next sections. The formalization of model-based assessment practice began in the early 60's, mainly with Fault Tree Analysis and Petri Nets. Petri Nets, in particular, have inspired many other formalisms (SPN, GSPN, etc.) which are currently widely used for model-based analyses. The largeness and complexity of the models rapidly became a challenging issue to be addressed, and beginning in the 80's researchers started focusing on the development of methodologies, techniques and tools to avoid or tolerate model complexity, also exploiting the synergies and complementarities among several evaluation methods (from the early 90's). Finally, in the last ten years, another research direction has focused on the use of engineering languages (UML, ADL, etc.) to facilitate the construction of the models for designers, and on the development of transformation techniques and tools to translate such high-level models to analysis models for dependability evaluation.

The remainder of the chapter is structured as follows. Section 9.1 provides a high-level view of the surveys concerning the model-based assessment topic that have been produced in some recent European projects. In Section 9.2 the available and commonly used modelling formalisms are outlined. Strategies to build and solve models in a time and space-efficient way are briefly discussed in Section 9.3, while Section 9.4 gives an overview of existing modelling and solution tools supporting model-based assessment. Section 9.5 discusses the methodologies used to derive dependability models based upon engineering models, and Section 9.6 surveys works that combine different modelling approaches. Some concluding remarks are then summarized in Section 9.7.

9.1. Surveys developed in recent European projects

Detailed surveys giving a comprehensive overview of model-based assessment can be found in recent European projects, discussed from different perspectives and applied to different system contexts.

In HIDENETS [HIDENETS 06], a state of the art of evaluation techniques, methodologies and tools is presented, focusing on distributed applications and mobility-aware services, in ubiquitous communication scenarios (see [Lollini 07]). Individual quantitative evaluation techniques that are suitable to analyze HIDENETS-like systems are reported, as well as existing work describing combinations of them.

In the context of ReSIST NoE [ReSIST 06a], the research challenge required for scalable resilience of policies, algorithms and mechanisms has been identified. In particular, the state of knowledge and ongoing research on methods and techniques for resilience evaluation are summarized in [ReSIST 06b], taking into account the scaling challenges of large and evolving systems considered in the project.

In CRUTIAL [CRUTIAL 06], a state of knowledge [Kaâniche 06] was presented, focusing on the existing model-based methodologies, techniques and tools that can be useful to address the challenges raised in the context of interdependent critical infrastructures in general, and electric power system infrastructures, in particular.

Finally, an overview on the model-based methodologies elaborated for the quantitative evaluation of safety critical applications, in particular of safety critical railway systems, has been produced in the SAFEDMI [SAFEDMI 06] project, and reported in [SAFEDMI 07].

9.2. Modelling formalisms

A system designer has in his or her possession a wide range of analytical modelling techniques to choose from. Each of these techniques has its own strengths and weaknesses in terms of accessibility, ease of construction, efficiency and accuracy of solution algorithms, and availability of supporting software tools. The most appropriate type of model depends upon the complexity of the system, the questions to be studied, the accuracy required, and the resources available to the study.

Analytical models can be broadly classified into non-state space (combinatorial) models and state space models.

9.2.1. Non-state space models

Reliability block diagrams (RBD's), fault trees (FT's) and reliability graphs (RG's) are non-state space models commonly used to study the dependability of systems. They are concise, easy to understand, and have efficient solution methods. However, realistic features such as interrelated behaviour of components, imperfect coverage, nonzero reconfiguration delays, and combination with performance can not be captured by these models. These arguments led to the development of new formalisms, such as dynamic fault trees (DFT's) and dynamic reliability block diagrams (DRBD's), to model reliability interactions among components or subsystems. A brief overview of traditional non-state space models can be found in [Nicol 04], while some of their 'dynamic' extensions are outlined in [Distefano 08].

9.2.2. State space models

State-space models, in particular homogeneous continuous time Markov chains (e.g., see [Trivedi 01] for full details) are commonly used for dependability modelling of computing systems. They are able to capture various functional and stochastic dependencies among components and allow evaluation of various measures related to dependability and performance (performability) based on the same model, when a reward structure is associated to them. Unfortunately not all the existing systems and their features can be captured properly by Markov processes; in some cases more general processes (e.g., semi-Markov, Markov Regenerative or even non-Markovian processes) must be used. When dealing with such processes, complex and costly analytical solution techniques may have to be used. If analytic solution methods do not exist, discrete-event simulation must be used to solve the models thus providing only estimates of the measures of interest. Especially for dependability metrics such as reliability and availability, simulation may however be time consuming because of the rare event problem: events of interest occur so rarely that very lengthy simulations are necessary to obtain reliable results. Alternatively, one can approximate an underlying non-Markovian process with a Markov process, and thus represent a non-exponential transition with an appropriate network of exponential ones (phased-type approach). The price to pay following this approach is a significant increase in the number of states of the resulting Markov model. The work in [Trivedi 96] reviews the existing state space models (as well as combinatorial ones) and it discusses the benefits and the limitations of each.

To facilitate the generation of state-space models based on Markov chains and their extensions, higher-level modelling formalisms like Stochastic Petri Nets (SPN's) are commonly used. These formalisms allow a more compact model representation because they

support concurrency. In [Ciardo 94], the authors explore and discuss a hierarchy of SPN classes where modelling power is reduced in exchange for an increasingly efficient solution, focusing on Generalized Stochastic Petri nets (GSPN's), Deterministic and Stochastic Petri Nets (DSPN's), Semi-Markovian Stochastic Petri Nets (SMSPN's), Timed Petri Nets (TPN's), and Generalized Timed Petri Nets (GTPN's). Other widely used modeling formalisms are Stochastic Reward Nets [Bobbio 98], Stochastic Activity Networks (SANs) [Sanders 01] and Markov Regenerative Stochastic Petri Nets (MRSPNs) [Choi 94].

9.3. Model construction and solution approaches

The main problem in using state-based models to realistically represent behaviour of a complex system is the explosion in the number of states (often referred to as state space explosion problem). Significant progress has been made in addressing the challenges raised by the large size of models both in the model construction and model solution phase, using a combination of techniques that can be categorized with respect to their purpose: largeness avoidance and largeness tolerance, see [Nicol 04], [Lollini 07] and [Kaaniche *et.al*. 2008] for three comprehensive surveys.

Largeness avoidance techniques try to circumvent the generation of large models using, for example, state truncation methods [Chen 02], state lumping techniques [Kemeney 60], hierarchical model solution methods [Sahner 96], fixed point iterations [Mainkar 96], hybrid models that judiciously combine different model types [Nelli 96] and the fluid flow approximation [Horton *et al.* 98, Hillston 05].

However, these techniques may not be sufficient as the resulting model may still be large. Thus, *largeness tolerance* techniques are needed to facilitate the generation and the solution of large state space models.

Largeness tolerance techniques propose new algorithms and/or data structures to reduce the space and time requirements of the model. This is usually achieved through the use of structured model composition approaches, where the basic idea is to build the system model from the composition of sub-models describing system components and their interactions. Generic rules are then defined for the elaboration of the sub-models and their interconnection. For example Multivalued Decision Diagram - MDD - data structures have been used in [Ciardo 99] to efficiently explore large state spaces.

Other techniques try to reduce the complexity of the model making use of concepts borrowed from the model checking theory. A first approach combines process algebras with Markov

chains, to take advantage of their powerful and well-defined composition operators, leading to the Input/Output Interactive Markov Chains (I/O-IMC) formalism [Boudali 07].

Rather than focusing on model composition, another approach concentrates on the definition of the dependability measures of interest to be evaluated. In fact, many sophisticated formalisms exist for specifying complex system behaviors, but methods for specifying performance and dependability variables remain quite primitive. To cope with this problem, modelers often must augment system models with extra state information and event types to support particular variables. To address this problem the so-called path-based reward variables have been introduced [Obal & Sanders 98]. Numerical methods to compute such reward variables, defined with the Continuous Stochastic Logic (CSL), are given in [Baier 99], while in [Haverkort 00] the model checking approach is illustrated through a workstations cluster example.

Other approaches try to tolerate model largeness using model decomposition and aggregation of the partial results. The basic idea is to decouple the model into simpler and more tractable sub-models, and the measures obtained from the solution of the sub-models are then aggregated to compute those concerning the overall model. A survey on decomposition/aggregation approaches can be found in [Lollini *et al.* 2009]. In the same paper, the authors also propose a general modeling framework that adopts three different types of decomposition techniques to deal with model complexity: at functional, temporal, and model-level. The key point is that the approach is non-domain-specific, i.e., not specifically developed for a particular class of systems or tailored for a specific modeling formalism or solution technique. Other largeness tolerance techniques also exist, such as disk-based approaches [Deavours 98a], where the model structure is stored in the disk thus allowing larger models to be solved, or on-the-fly approaches [Deavours 98b] which completely avoid the storage of structures in memory, generating them iteratively while computing the solution.

It is important to note that largeness avoidance and largeness tolerance are complementary and both are needed at the model construction and model solution levels, when detailed and large dependability models need to be generated and processed to evaluate metrics characterizing the resilience of real life systems.

9.4. Modelling and solution tools

Several software tools developed over the last thirty years address dependability and performability modelling and evaluation. Surveys of the problems related to techniques and tools for dependability and performance evaluation can be found for example in [Haverkort 96, Reibman 91, Sanders 99]. Tools can be grouped in two main classes:

- *Single-formalism/multi-solution tools*, which are built around a single formalism and one or more solution techniques. They are very useful inside a specific domain, but their major limitation is that all parts of a model must be built in the single formalism supported by the tool. In the following we cite two sets of tools. The first set of tools is based on Stochastic Petri Nets formalism and its extensions. They all provide analytic/numerical solution of a generated state-level representation and, in some cases, support simulation-based solution as well. This set includes DSPNexpress [Lindemann et al. 99], GreatSPN [Chiola et al. 95], SURF-2 [Béounes et al. 93], DEEM [Bondavalli et al. 00], TimeNET [German et al. 95], UltraSAN [Sanders et al. 95]. Another set of tools uses other model specification approaches, sometimes tailored to a particular application domain, and it includes HIMAP [Sridharan et al. 98] and TANGRAM-II [Carmo et al. 97].
- *Multi-formalism/multi-solution tools*, which support multiple modelling formalisms, multiple model solution methods, and several ways to combine the models, also expressed in different formalisms. They can be distinguished with respect to the level of integration between formalisms and solution methods they provide. In particular, some tools try to unify several different single-formalism modeling tools into a unique software environment. Examples are the following: IMSE [Pooley 91], IDEAS [Fricks et al. 97], FREUD [van Moorsel & Huang 98], DRAWNET++ [Franceschinis et al. 02]. In other tools, new formalisms, composition operators and solvers are actually implemented within a unique comprehensive tool. Though more difficult than building a software environment out of existing tools, this approach has the potential to much more closely integrate models expressed in different modeling formalisms. To the best of our knowledge, there are five main tools having these attributes: SHARPE [Trivedi 02], SMART [Ciardo & Miner 96], DEDS [Bause et al. 98], POEMS [Adve et al. 00] and MÖBIUS [Daly et al. 00].

The solution is considered as a computation of a measure by using one of the following classes of techniques:

- *Closed form results*, which yield exact measures but can be obtained for only a limited class of models.
- *Direct analytical techniques*, like matrix inversion, which still yield exact measures but can be obtained for only a limited class of models.
- *Iterative numerical techniques* (to note that no general guarantees of convergence of iterative methods do exist for some problems, and the determination of a suitable error bound for termination is not easy).
- *Simulation techniques*, which provide an estimate with a confidence interval for the result, but may be costly in terms of run time.

In a large number of scenarios, steady-state solutions by itself are insufficient. Two prominent examples for that are (1) scenarios in which steady-state does not exist at all, and (2) systems that can only be described by a homogeneous model for a very limited time during which steady-state behaviour cannot be observed. Transient analysis is frequently performed in simulation models although there exists a number of methods (supported by several tools) for analytical models as well.

9.5. Deriving dependability models from engineering models

The emergence of model-based engineering methodologies and the elaboration of automated model transformation techniques have opened up new ways to integrate model-based assessment into the development process. Model-based engineering refers to the systematic use of models as primary artefacts throughout the engineering lifecycle [Schmidt 06]. Precise, albeit informal, engineering languages (like UML, BPEL, AADL, etc.) allow not only a reasonable unambiguous specification and design but also serve as the input for subsequent development steps like code generation, formal verification, and testing. One of the core technologies supporting model-based engineering is *model transformation* [Czarnecki 03]. Transformations can be used to refine models, apply design patterns, and project design models to various mathematical analysis domains in a precise and automated way. Graph transformation techniques and the related tools are considered as the most important achievements in this area [Baresi 02].

These initiatives and technologies influenced model-based assessment as well, since they offered an efficient and integrated approach to *derive dependability analysis models from engineering models*. Resilience assessment is a specific task in model-based engineering since

in this case so-called *non-functional aspects* (like reliability, safety) are addressed that are not properly covered by the common engineering languages (these focus primarily on functional aspects). Accordingly, language extensions like the UML profiles for QoS and fault tolerance, BPEL extensions, the Error Annex for AADL, etc. were introduced and utilized.

Two approaches for deriving dependability analysis models have appeared in practice:

- System designers use the extended engineering language to directly describe failure and repair/recovery related behaviour and properties of components, and an underlying toolset constructs models and computes system-level dependability measures [Ganesh 02].
- Dependability experts construct *analysis sub-models* that represent the failure and recovery processes of the different types of components and the error propagation among them. System designers use the language extensions just to identify the component types and assign local dependability parameters to hardware and software artefacts in the engineering model. Automated tools (based mainly on pattern matching and model transformation) are then used to (i) assemble in a modular way the relevant sub-models on the basis of the system structure and (ii) invoke solution algorithms to solve the system level model. Examples of this approach are related to UML [Bondavalli 01, Kovacs *et al.* 2008], AADL [Rugina 07] and business process models [Gönczy 06]. The analysis models used in these techniques are Markov chains, GSPNs and Stochastic Reward Nets.

9.6. Works combining different evaluation approaches

The largeness and complexity of current real-life systems call for a *composite V&V* (verification and validation) framework, where the synergies and complementarities among several evaluation methods can be fruitfully exploited. For example, it is well established and widely recognized that modelling and experimentation complement each other, at least at the conceptual level, but the two approaches are infrequently combined in the literature to evaluate real-life systems.

An interesting area of research is the construction of analysis models *on the basis of measurements* performed in a running prototype or in a full deployment. The most comprehensive method was developed for performance and performability analysis: software performance models of distributed applications are extracted from traces recorded during execution [Israr 07]. A similar approach is recording error propagation traces induced by fault injection experiments [Arlat 93] to support the construction of error propagation models [Chillarege 89]. Other works (e.g., [Arlat 90]) derive high-level behavioral models using

experimental measurements obtained from fault injection experiments, while in other papers (e.g., [Coccoli 02]) the values provided from field data are used to setup parameters of analytical models.

Other attempts in exploiting the potential interactions among different evaluation approaches were reported in the context of two European projects: DBench [DBench] and HIDENETS [HIDENETS 06]. In DBench, a framework for dependability benchmarking based on modelling and experimentation was defined, focusing on On-Line Transactional systems (see [Kanoun 04] for more details). In HIDENETS, a holistic approach for the analysis of distributed applications and mobile services on dynamic and open communications infrastructures has been presented. In this context, several (complementary) evaluation techniques have been used in cooperation to reach a common objective, that is the end-to-end quantitative assessment of dependability and QoS indicators, both as system's characteristics and as perceived QoS of the services offered to the users (see [Lollini 07] for more details). In [Bondavalli *et al.* 2009], for example, a Stochastic Activity Network modelling approach has been combined (integrated) with an ad-hoc mobility simulator that allows to capture system characteristics at a more detailed level, thus enabling a more refined analysis that could be hardly obtained using a single technique.

To the best of our knowledge, there are no surveys on works combining different evaluation approaches.

9.7. Open research challenges

As discussed in this chapter, a wide literature exists on the use of models for the assessment of dependability-related indicators of a system and, in general, for fault-forecasting, that is to probabilistically estimate the occurrence of faults and their impact on the ability of the system to provide a proper service. Nevertheless, the modelling and analysis of complex (large, dynamic, heterogeneous, ubiquitous) systems still needs continued research, both in model construction and in model solution. A crucial point in this context is also to assess the *approximations* introduced in the modelling and solution process to manage the system complexity, as well as their impact on the final results.

The role of modelling in a more comprehensive assessment process is, on the contrary, not well addressed in the literature. The largeness, dynamicity, heterogeneity and ubiquity of current computing systems actually calls for the development of a *composite and trustable assessment framework* including complementary evaluation techniques, covering modelling

and experimental measurements. Mechanisms are needed to ensure the cooperation and the integration of these techniques, in order to provide realistic assessments of architectural solutions and of systems in their operational environments.

Of increased significance is also the use of quantitative evaluation methods to support the effective use of *adaptation mechanisms* in current systems. Efficient on-line mechanisms are needed to monitor the environment conditions of the system and to dynamically adapt to their changes.

Besides assessing the impact accidental threats, extensions are also needed to *quantify the impact of malicious threats*. Quantitative evaluation techniques have been mainly used to evaluate the impact of accidental faults on systems dependability, while the evaluation of security has been mainly based on qualitative evaluation criteria. Therefore, there is a need for a comprehensive modelling framework that can be used to assess the impact of accidental faults as well as malicious threats in an integrated way.

10. Security Assessment

As more and more tasks in daily life are processed using computer systems and networks, confidentiality of information and security of the whole system is becoming increasingly vital. The traditional aspects of security are confidentiality, integrity and availability ("CIA") [ITSEC 1991]. See also [Avizienis 04] for a comprehensive discussion of the subject. While availability of systems has traditionally been an important component of dependability, the other two properties have more recently gained increasing research attention. Integrity has been mostly addressed in distributed system research and confidentiality is the focus of much of today's research in the area of security. At the same time there exist different approaches, which try to integrate security and dependability attributes into a common framework [Avizienis 04] [Jonsson 2006].

Security is a very active field of research with many open questions. Some of this research goes beyond exploring and developing methods for security, and ventures into measuring the degree of security a system provides and evaluating how secure a certain system design will be under certain operational conditions.

This chapter surveys existing work on metrics for security and methods and tools to measure and assess security. This area has been recognized as increasingly important. For instance, Chief Information Security Officers representing over twenty Fortune 500 firms listed security metrics first among six key security imperatives for large enterprises. The US Information Security (INFOSEC) Research Council poses it on their hard problem list as follows: 'The objective is to create a scientific foundation, methods and tools for quantitative assessment of measures of combined security, safety and reliability (...)' [I3P] Metrics are thus critical for industry, government, research programmes, and operators alike [Sanders et al 2006].

10.1. Measuring the Past versus Predicting the Future

In the area of security, the uses of measurement and more generally of quantitative assessment are controversial and, according to most authors, largely immature. The problems start with the very choice of the attributes to be measured. In security, the practical difference between *measuring of the past* and *prediction* is especially striking.

Agreeing measures of *how secure a system has been*, e.g., how many penetrations of a certain type have occurred on certain installations over a certain time interval is conceptually simple;

their practical measurement depends on the availability of trustworthy enough detection and recording tools. However, assessment ideally is mostly about *how secure the system will be*, and here the measure obtained for the past is more obviously inadequate than for most measures outside security. The attack types and the effort spent by attackers in any future time interval will usually be different from the past. To consider an extreme but not necessarily unusual scenario, the very fact of publishing measures of past security might cause them to change.

The difficulty of such predictive assessment leads many practitioners and researchers to dismiss the very idea of quantitative assessment of security, while others (including contributors to this document) insist on the need for a sound definition of these quantitative measures as a basis for sound reasoning, irrespective of the practical difficulties in measuring them.

10.2. Metrics to Assess Security

A primary challenge in security assessment is the identification of a metric that represents security or one of its facets appropriately. The metrics must be quantitative, measurable, scalable, useful, and effective for systems under consideration [Sanders et al. 2006]. We discuss formal characterisation of security properties, qualitative/descriptive methods, strictly quantitative assessment and security economic metrics and concerns.

Formal Properties. There exists a rich history of the application of formal methods to security problems, see e.g., [Wing 1998] for a historical survey. Security properties can be formulated as Boolean properties (of systems) that can be either true or false. This perspective lends itself for formal verification of correctness, as well as pseudo-formal proofs (by precise reasoning). It is thus useful to developers in designing and verifying properties of algorithms or implementations. One could for instance demonstrate (under certain assumptions) that a user of a computer will not be able to perform a certain operation without first entering a correct password. A workshop on Formal Methods in Security Engineering exists now in its 7^{th} year.

However, for the purpose of assessing security properties (e.g., privacy properties) of whole systems in operation, such a perspective by its nature makes restrictive, possibly unrealistic, assumptions about users, attackers, system operation and other interactions with the system. Quantification would be needed to assess the coverage of the assumptions underlying the formal model. In the real world, practically no system is absolutely secure, so a quantitative

notion of “degree of security” is needed. The formal approach also defines security purely in terms of a system property, ignoring other relevant aspects such as the scenario in which the system is used, and measures such as business impact of IT security decisions. In terms of measurement theory, we would say that operational security is an *external* attribute of systems (dependent, that is, on circumstances outside the system), while the properties that can be verified are usually *internal* ones (easier to capture in objective ways but of more limited utility).

Descriptive Methods. Most practical “security assessment” activity tends to rely on descriptive methods, checking for the presence of specified security mechanisms in design and operation and possibly rough indicators of their quality [Jaquith 2007], [Payne 2006]. Such classifications are sometimes mapped onto ordinal measurement scales from “less secure” to “more secure” (according to one or more security attributes) for ranking different systems. But these “metrics” are just descriptions of what has been done to achieve security in operation; they are not proven to lead to trustworthy predictions of security properties in operation; the rankings they determine are not necessarily predictive of rankings in terms the values of such measures. It should be noted that it is enormously challenging to validate scientifically if the proposed metrics are sufficiently expressive and predictive. To the best of our knowledge, no scientific validation approaches for security metrics have been proposed in the literature.

The most commonly used approach to descriptive metrics is based on the assumption that a proper design and development process would be sufficient to prevent from malicious threats. Such approaches resemble approaches to safety certification [Brosgol 2008]. Accordingly, several security evaluation criteria have been developed to assess the security of computer based systems and compare their ability to cope with malicious faults, e.g., TCSEC [TCSEC 1985], ITSEC [ITSEC 1991] and the common criteria [CC 2006]. These criteria are qualitative and take into account the functional aspects of the system and the methods used in the development process. With respect to risk assessment approaches, the goal is to analyse and determine the likelihood that identifiable threats will affect the target system security, weighing their occurrence with the damage they might cause. Damage is generally assessed by evaluating the average loss of money an attack might cause.

NIST [Swanson et al. 2003] provides guidelines for IT security officers to develop and implement an IT security metric programme, and various other groups have built on this. For instance, Corporate Information Security Working Group [CISWG 2004] expands on the

NIST guidelines by listing metrics that represent the percentage of systems, procedures and personnel confirming to best practice. Analogous to the maturity model for software development, there also exist maturity models such as the Systems Security Engineering Capability Maturity Model [CMM 2003]. These express the maturity of development and deployment processes and personnel with respect to security criteria. Although many metrics have thus been proposed, most of them relate to processes instead of deployed systems. Although this can be considered a weakness, it illustrates the challenge to assess the security of deployed or designed systems and define useful metrics for such security assessment,

Quantitative Measures. Alternatively, one assesses security in terms of quantitative measures taking into consideration these complicating factors using probabilistic models. Some researchers have attempted to clarify the use of probabilities in the context of security [Littlewood *et al.* 1993]. A more recent discussion of the value and difficulties of probabilistic assessment can be found in [Littlewood & Strigini 2004], which also underscores the need for precise characterisation of the attributes considered, for any use of such measures to be useful.

For particular security aspects, metrics can be meaningfully designed to evaluate the security of the particular mechanism. The most well-known example can be found in quantification of the effort required to break cryptographic algorithms. Another example was introduced by [Diaz & Preneel 2004], who propose a probabilistic metric of anonymity for email mixes defined by the mixes' entropy. Such approaches are particularly valuable, but assess the security only within the context of the metric. That is, IT professionals or researchers of other aspects of system design or security will still find it challenging to translate such metrics into ones that help them make decisions about security for their companies or in their designs.

Security Economics. An important new strand of research assesses security in light of the economic forces that influence real-life security decisions [Anderson & Moore 2007, Shostack & Stewart 2008]. The Workshop series on Economics of Information Security (WEIS) is set in this field. As an example, liability for fraud pressure banks to secure their web sites, while the lack of such liability discourages security investments [Anderson & Moore 2007]. In such cases decisions about security are primarily business decisions, not isolated technology decisions. These decisions can only be properly understood when identifying the economic and business arguments and require assessment that goes beyond technology.

In similar vain, recent research to improve enterprise security decision making stresses the importance of understanding business concerns as well as human factors. For example, the

work on trust economics [Coles et al. 2008] includes user studies to predict the way users will react to new security investments as well as estimates of business impact. Mathematical models trade off business, security and human concerns, such as productivity versus risk of data loss [Parkin et al. 2008] or establish thresholds for individual's willingness to comply with security policies [Beautement et al. 2008].

10.3. Assessment Approaches

Model-Based Quantification. Particular attention has been paid during the last decade to exploring new approaches for security evaluation based on probabilistic models similar to those used in computer dependability. Studies carried out in the nineties develop representative quantitative measures characterizing the capacity of a system to resist to attacks in operation and probabilistic methods for security evaluation analogous to those used for the evaluation of reliability (see e.g., [Littlewood *et al.* 1993] [Dacier *et al.* 1996] and [Ortalo *et al.* 1999]). This work introduced the concept of *effort* to derive quantitative security metrics and proposed stochastic models for their evaluation. Similarly, the work in [Brocklehurst et al. 1994], [Jonsson & Olovsson 1997] and [Littlewood & Strigini 2004] uses the concept of attacker effort (until a security breach) to define quantitative security metrics, analogously to reliability metrics that are defined in terms of time to failure, and proposed stochastic models for their evaluation. [AlTurki 2009] uses a probabilistic logic to model DoS and protocols preventing this.

Related approaches have emerged more recently, see the survey presented in [Nicol 04]. We mention in particular the state-based models proposed in [Madan *et al.* 2002, Gupta *et al.* 2003] targeting intrusion-tolerant systems. Other types of stochastic models (also surveyed in [ReSIST 07]) have been explored in the context of security. These include e.g., epidemiology models or interactive Markov chains for studying malware propagations (see e.g., [Staniford *et al.* 2002, Garetto *et al.* 2003, Zou *et al.* 2005], models based on complex network theory to analyse the vulnerability of networked systems to cascading attacks [Crucitti *et al.* 2004], game theory-based models for characterizing attackers behaviours and analysing impact on security [Lye & Wing 2005, Sallhammar *et al.* 2006].

Note that the above model-based approaches assess security by describing separately and in detail the rules that govern the behaviour of attackers and of system defences and to play out the possible consequences of their interaction. But this requires first empirical knowledge of how attackers actually behave, as well, often, a statistical characterisation of the strength of

the defence mechanisms. That is, the models do not rely on intended deterministic reactions to specific events, but on the probabilities of it reacting correctly, or not, over large populations of such events, taking into account any weakness in the specification or implementation of the mechanism [ReSIST 07].

Therefore experimental data is required to determine these input parameters, as we discuss below. As examples of the interplay between measurement and modelling, based on statistical analysis of experimental data a stochastic model for changing passwords has been set up and parameterised [Cukier 2007]. Initial exploratory models illustrating various types of empirical analyses can be found in [Bloomfield *et. al.* 2008], [Kaaniche *et. al*. 2006].

Experimental Quantification and Honeypots. Experimental studies have emerged for collecting and analysing real data characterizing attacks and malicious activities on the internet, see for example [Bailey *et al.* 2005], [Cukier et al. 2006],[Kohno *et. al.* 2004],[Meyer 2006],[Cymru 2004], [CAIDA], [DShield] and [Leurre.com 2008]. In this light, we refer to the Field Measurement chapter in this document for a survey of security field data collected for web servers. Such data is of interest in itself, but, as we explained above, is also necessary to understand attacker behaviour and to elaborate realistic assumptions about malicious threats and attackers behaviours.

One prevalent method of studying attacker behaviour is by using so-called honeypots or honeynets, i.e. dedicated systems on which security breaches can be observed [Bloomfield *et. al.* 2008] [HoneynetProject 2008] [Leurre.com 2008] [Spitzner 2003]. In particular research networks of honeypots provide a rich source of data, and numerous honeypot systems have appeared [ODP]. Recent work focuses on advanced analysis methods to deal with the large amounts of data collected using the honeypots [Thonnard 2008], and on further advances in the honeynet technology itself to increase the attacker interaction paradigms one can observe (advancing further on the roadmap given in [Spitzner 2003]).

Intrusion Detection. Intrusion detection reflects an important category of technologies that relates to a subset of security assessment in its need to assess at run-time if an attack is carried out. This is different from security assessment in general in that it does not determine a degree of security of a system, neither measuring it, nor attempting to predict it. The efficiency of an intrusion detector can be determined in terms of false positives, false negatives and success rate. More specifically, intrusion detection is either anomaly detection or misuse detection [Anderson 2008]. Anomaly detection uses knowledge of normal system behaviour which, together with system specification, is used to build a profile of normal operation. Patterns that

deviate from known healthy system behaviour are indicators for system misuse or attack. In contrast, misuse detection is equipped with a set of attack descriptors (or signatures) and tries to match series of new events with known patterns of misuse. This method has the disadvantage of not being able to detect new types of attacks. An advantage is that it usually produces very few false alarms.

For both, anomaly and misuse detection the system is monitored and an *audit trail* of relevant system parameters is created. An important task in intrusion detection is the analysis of the audit trail, the *sensor alert correlation*. An example of an experimental evaluation of sensor alert correlation is in [Haines *et al.* 2003]. A testbed of 59 hosts, divided into four sites with each one to two local networks. The testbed is equipped with a number of network-based as well as host-based sensors and firewalls. The sensors produce alerts and patterns in the alert sequences are analysed to detect an attack. [Valeur *et al.* 2004] combine previous work on sensor alert correlation in a framework and apply it to experimental data. Alert correlation is used to reduce the amount of data to be studied by the system operator and to combine sensor data into aggregate attack notification. [Haines *et al.* 2003] apply correlation to several data sets and find that the success of the correlation process very much depends on the data set.

Red Teaming. The strength of defences is studied in penetration testing using artificial attacks. These attacks are performed by external teams of security experts ("red teams" or "tiger teams"), such as Sandia IDART, that aim to achieve specified goals in breaching the security of the system [Nichols 2007, KL 2001, Kraemer 2004]. Attempts to measure adversary work factors via red teaming are the closest attempt to date to measuring whole-system security empirically. Despite the practical importance of direct and realistic empirical data, they clearly raise problems of representativeness (is the threat presented by the red team representative of the real-world threats), and thus of predictive value of the measures obtained, and of how to derive essentially probabilistic measures from the few data points affordable.

Security Testing. Security testing of networks has been based on the open source security testing methodology manual (OSSTMM) [Corral 2005]. Several approaches to automating penetration testing also exist, e.g. systems like SATAN [Farmer 2002], MetaSploit [Ramirez-Silva & Dacier 2007] and the NESSUS tool which was originally designed for aircraft security and reliability testing [Riha *et. al.* 2000]. Also important is the new development of attack injection, which is the security equivalent of fault injection [Neves et al. 2006].

10.4. History

Security assessment and evaluation is a relatively new research field and many open questions remain. [Anderson 2008] defines security assurance and evaluation as among the most relevant challenges for the coming years. Important industry-driven work exists, such as detailed in [Jaquith 2007] and discussed in securitymetrics.org, but systemic mathematically powerful assessment methods remain illusive. Since the early nineties model checking and theorem proving have been applied to protocols for secure key transmission and authentication [Wing 1998]. This, however, provides a binary outcome about the protocol correctness and the need for quantitative assessment of security was therefore already formulated in [Littlewood *et al* 1993. This decade has seen probabilistic analysis taken off even though it still is immature and at an early stage.

Security measures have been defined and experimental methods of evaluation have been applied [Ortalo *et al.* 1999, Haines *et al.* 2003]. Model-based evaluation of a system (as opposed to its development or deployment process) is only just starting [Zhao 2008, Thomas 2005] and can be expected to become increasingly important in the future.

11. Safety and Assurance Cases

This chapter reviews approaches used in safety cases for software based systems. To do this it is important to understand the system safety and regulatory context that is introduced in Section 11.1. This leads into the current uptake in safety and assurance cases and the current practice on structured safety cases is briefly described. The chapter provides an explanation of the historical development of cases and provides some pointers to literature and examples and lists some areas of research (perhaps to be developed later in road map document).

11.1. Background to software safety and regulation

11.1.1. Introduction

In this section we provide background to regulation and safety cases. This draws heavily on earlier work: regulations from CAA [CAA 98][Penny et al 01] and the NuSAC SOCS report [HMSO 86], which also provides more extensive background to why software safety is difficult to achieve and a review of practices in a number of countries and sectors. Other background material is available from the IEE briefing material that we produced [Adelard]. We also elaborate a brief history of software safety[5] standards and safety cases to provide an additional perspective for the review.

11.1.2. Approaches to regulation

The justification that a system is fit for purpose (and continues to be fit for purpose as the environment, use and implementation change) is a complex socio-technical process. In safety regulation in general there has been a widespread adoption of safety case regimes. The Robens Report [Robens 72] and the Cullen Enquiry [Cullen 90] were major drivers behind the UK Regulatory agencies exploring the benefits of introducing goal-based regulations. The reports noted several shortcomings with prescriptive safety regulations: that is regulations that provide a strict definition of how to achieve the desired outcome.

Firstly, with prescriptive regulations, the service provider is only required to carry out the mandated actions to discharge his legal responsibilities. If these actions then prove to be insufficient to prevent a subsequent accident, it is the regulations and those that set them that are seen to be deficient. Thus safety can be viewed as the responsibility of the regulator and not the service provider whose responsibility, in law, it actually is.

[5] We use software safety standards as shorthand for software standards that apply to software in a system that has a safety role. Of course safety is an overall system property not that of the software.

Secondly, prescriptive regulations tend to be a distillation of past experience and, as such, can prove to be inappropriate or at worst to create unnecessary dangers in industries that are technically innovative.

Thirdly, prescriptive regulations encode the best engineering practice at the time that they were written and rapidly become deficient where best practice is changing e.g. with evolving technologies. In fact it is quite probable that prescriptive regulations eventually prevent the service provider from adopting current best practice.

Another driver for adopting goal-based regulation, from a legal viewpoint, is that overly-restrictive regulation may be viewed as a barrier to open markets. Various international agreements, EC Directives and Regulations are intended to promote open markets and equivalent safety across nations. Whilst it is necessary to prescribe interoperability requirements and minimum levels of safety, prescription in other areas would defeat the aim of facilitating open markets and competition.

Finally, from a commercial viewpoint, prescriptive regulations could affect the cost and technical quality of available solutions provided by commercial suppliers. So there can be clear benefits in adopting a goal-based approach as it gives greater freedom in developing technical solutions and accommodating different standards.

A system safety case is now a requirement in many safety standards and regulations. Explicit safety cases are required for military systems, the off shore oil industry, rail transport and the nuclear industry. For example, in the UK a nuclear safety case must demonstrate, by one or other means, the achievement of ALARP. In the Health and Safety Commission's submission to the Government's 'Nuclear Review'[6] a Safety Case is defined as 'a suite of documents providing a written demonstration that risks have been reduced to ALARP. It is intended to be a living dossier which underpins every safety-related decision made by the licensee.'

The system safety case of course varies from sector to sector. The core of a nuclear system safety case is (i) a deterministic analysis of the hazards and faults which could arise and cause injury, disability or loss of life from the plant either on or off the site, and (ii) a demonstration of the sufficiencies and adequacies of the provisions (engineering and procedural) for ensuring that the combined frequencies of such events will be acceptably low. Safety systems will feature amongst the risk reducing provisions comprised in this demonstration, which will thus include qualitative substantiations of compliance with appropriate safety engineering

[6] The review of the future of nuclear power in the UK's electricity supply industry.

standards supplemented (where practicable) by probabilistic analyses of their reliabilities. Other techniques which may be used for structuring the safety case include fault and event tree analysis, failure mode and effects analysis (FMEA) and hazard and operability studies (HAZOPS).

The safety case, particularly for computer based systems, traditionally contains diverse arguments that support its claims. These arguments are sometimes called the 'legs' of the safety case and are based on different evidence. Just as there is defence in depth in employing diversity at system architecture level, so we see an analogous approach within the safety case itself. Another important feature of the safety case process is independent assessment. The objective of independent assessment is to ensure that more than one person or team sees the evidence so as to overcome possible conflicts of interest and blinkered views that may arise from a single assessment. The existence of an independent assessor can also motivate the assessed organisation. The relationship between independent assessment and "legs" can however be complex.

Safety cases are important to minimise safety risks as well as commercial and project risks. In industries such as the nuclear industry, the need to demonstrate safety to a regulator can be a major commercial risk.

So to sum up the motivation for a safety case is to:

- provide an assurance viewpoint that demonstrates that safety properties are satisfied and risks have been satisfactorily mitigated
- provide a mechanism for efficient review and the involvement of all stakeholders
- provide a focus and rationale for safety activities
- demonstrate discharge duty to public and shareholders
- allow interworking between different standards and support innovation

So in a safety case the emphasis should be on the behaviour of product not just the process used to develop it: a useful slogan is "What has been achieved not how hard you have tried".

11.1.3. A brief history of software safety cases and relevant standards

The nuclear industry has had a major influence on the development of approaches to safety related computer system development and assurance. From the late 1970s the European Working Group on Industrial Computer Systems, a cross sector pre-standardisation working

group, developed a series of guidelines and books that documented best practices. The guidance led to a large part of IEC 880. In the early 1980s, developments in Italy and the Sizewell B proposals in the UK provided significant contributions to that work. The experience of EDF and Merlin-Gerin with the first generation of reactor protection systems, SPIN, was fed into the committee. The software engineering approach in the EWICS guidelines [Redmill][Robens 72] and Peter Bishop's book represented the state of the art at that time [Bishop and Bloomfield 95].

In the 1980s there was also a nuclear industry PES Working Group that led to an early version of the PES guidelines and input to Design Safety Guidelines. It considered some enduring themes: the need for a common mode cut-off (the study by [Bourne et al 81] being very influential and one of the few sources of CMF data) and development guidance. The PES Working Group had representation from the electricity and nuclear industry, CAA and MoD.

In the UK there were a number of policy initiatives. The ACARD report HMSO 86] in 1986 and a subsequent IEE/BCS and HSE studies IEE 89] UKHS 87] set the scene and in 1988 the Interdepartmental Committee on Software Engineering (ICSE) established its Safety-Related Software (SRS) Working Group to coordinate the Government's approach to this important issue. Members were drawn from a wide range of departments and agencies: CAA, CEGB, DES, DoE, DTI, DoH, DoT, HSE, MoD, RSRE, and SERC. The work was motivated "not by a recognition of particular present dangers; rather by a desire to anticipate and forestall hazards which may arise with the very rapid pace of technical change."

The UK HSE were active in taking the lead in the UK Governmental Interdepartmental Committee on Software Engineering (ICSE) and this, with support from DTI, led to a consultation document known as SafeIT [Bloomfield 90] and an associate standards framework Bloomfield and Brazendale 90]. HSE also published awareness documents (that led to Out of Control [UKHS 93]) on safety and PES (as they became known) and some earlier studies that looked at the feasibility of providing a validated framework for selecting software engineering techniques. SafeIT identified four main areas of activity requiring a coordinated approach: standards and certification; research and development; technology transfer; education and training.

The UK MoD were, as one might expect, pioneers in the use of critical software and the development of static analysis tools to analyse the code (Malpas) as well as forays into formally proven hardware designs. In the light of finding defects in certain operational systems, dramatic changes to the supply chain as well as reductions in MoD scientific

personnel, they responded in 1989 with the publication of a new draft interim standard 00-55 [UKMoD 89]. This used expertise from the nuclear and aerospace industry, MoD and elsewhere to develop a market leading standard around the requirements for mathematically formally verified software and statistical testing. It was soon realised – in part because of the attempt to classify systems as non-safety critical and outside the remit of 00-55 – that a wider system standard was needed. This led to Def Stan 00-56 [UKMoD 91]. There was considerable work to take into account the strong industry and trade association comments (led by the DTI that developed a detailed trace from all comments to the final issue of the standard).

In parallel with the development in the defence sector, the HSE led the production of the IEC generic standards that became known as IEC 61508 [IEC 98]. Draft publications [IEC 93] emerged in the early 1990s sharing much in common with the defence standards but addressing a wide range of systems and safety criticalities. During their prolonged drafting they developed detail, consistency and international recognition. However the technical basis of their software aspects remained fixed. The software techniques guidance in IEC 61508 and its software engineering approach was essentially just an extension and internationalisation of the work of [Bishop and Bloomfield 95]. There are a number of technical difficulties in how Safety Integrity Levels (SILs) are used and there is a lack of a requirement for safety cases.

In the nuclear industry the licensing of the Sizewell B Primary Protection System (PPS) had a major impact in terms of the visibility of the subject, the technical rigour required by the nuclear industry (the 250 person years of effort on independent assessment, largely static analysis based on Malpas but also subsequent statistical testing) and the impetus it gave to defining and articulating the special case licensing approach. This resulted in a number of public papers (e.g. [Hughes and Hall 92][Hunns and Wainwright 91] and the SAPS. In 1991-3 there were a number of Nusac papers on the software case, the claims and the negotiations on the PPS. Around 1993 the limitations to the claims that could be justified by testing were investigated by Butler and Finelli at NASA [Butler and Finelli 93], and similar results, involving testing and other evidence, by Littlewood and Strigini [Littlewood and Strigini 93]. The 10^{-4} limit was one set by pragmatics of testing technology, but did not include the assumption of doubt that we might now make explicit [Bloomfield and Littlewood 07].

In 1997 the 1991 Interim MoD Def Stan 00-55 was revised to become a full standard and became the first standard to explicitly require a software safety case. This was a radical departure from previous standards but offered some flexibility in the justification of the

software, important in view of industry comments on the interim standard. The key features of the revised standard were:

- Deterministic reasoning and proof
- Statistical testing
- Importance of range of attributes (not just correctness)
- Multi-legged arguments and associated metaphors (belts and braces rather than a wing and a prayer)
- Safety cases and reports
- Sound process to provide trustworthy evidence
- Systematic approach and clarity of roles and responsibilities and other recommendations to reduce project risks
- Evidence preferences: *deterministic* evidence is usually to be preferred to statistical; *quantitative* evidence is usually to be preferred to qualitative; *direct* evidence is usually to be preferred to indirect

The nuclear expertise was influential in Def Stan 00-55. As with many standards and guidelines 00-55 grappled with how to treat software of lower criticalities: at one extreme everything is required and at the other a minimum set of good practices. Populating the regions in between has been problematic and largely a product of the standards process rather than the scientific one. More recently 00-55 has become part of a reissued 00-56 [UKMoD 04] and no longer contains software integrity levels.

Adelard had an important role in the development of the defence standards and drafted the safety case requirements. The origins of the work go back to the individuals' involvement (Bloomfield, Bishop, Froome) in the days of the Public Inquiry into the Sizewell B Primary Protection System [Sizewell 82]. The work was later described in the concepts of Toulmin [Smith and Stockham 07] although developed somewhat independently. The concepts were first documented in the EU SHIP project [Bishop and Bloomfield 98][Bishop and Bloomfield 95] and the work was taken up within the IMC QUARC project. This led to the first software safety case publication and, in 1998, to ASCAD [Bloomfield et al 98][Bishop and Bloomfield 98], a safety case development manual (still the only one). ASCAD provided the now customary definition of a case as "a documented body of evidence that provides a convincing

and valid argument that a system is adequately safe for a given application in a given environment". In addition to the Adelard work there was research being done at York University [McDermid 94] that later led to the Goal Structuring Notation being described in Kelly's 1998 PhD [Kelly 98].

The ASCAD manual incorporated, with permission, considerable work from the UK nuclear research programme:

- On long-term and safety case maintenance
- How to address specific design issues, even the work on reversible computing
- The work on worst case reliability bounds
- Field experience collected from a range of projects and also used in the SOCS report [HMSO 86]
- On argument architecture based on analogies and analysis of PWR pressure vessel cases [Hunns and Wainwright 91][Sizewell 82]

It also made use of nuclear work on safety culture and work from REAIMS on organisational learning and human factors [Bloomfield et al 98].

In 1995 the Advisory Committee on the Safety of Nuclear Installations (ACSNI)[7] set up the Study Group on the Safety of Operational Computer Systems with the following terms of reference:

- to review the current and potential uses of computer systems in safety-critical applications;
- to consider the implications for the nuclear industry;
- in this context, to consider developments in the design and safety assessment of such computer-based systems, including other aspects of control systems;
- and to advise ACSNI where further research is necessary.

The report from this group [Littlewood 98] addressed the broad principles upon which the evidence and reasoning of an acceptable safety case for a computer-based, safety-critical system should be based. It also discussed, but did not attempt to cover in detail, the extent to which the UK nuclear industry already accepts these principles in theory, and the extent to

[7] Became the Nuclear Safety Advisory Committee (NuSAC) and then disbanded.

which they act on them in practice. It made a number of recommendations on regulatory practice, safety cases, computer system design and software engineering, standards, and research.

11.2. Uptake and development of the safety case approach

The incorporation of software safety case requirements in the defence standards drove interest in safety cases, and other forms of assurance case. A generalisation of the safety case concept also appears in Def Stan 00-42 Part 3, on the Reliability and Maintainability Case, and Part 2, which deals with the software reliability case. Similar requirements appear in equivalent NATO standards. Adelard has developed and marketed a supporting tool for safety cases - ASCE – and published a supporting methodology in the ASCAD manual. Much of the work on safety cases and the supporting research is not published and this is becoming increasing an issue. By their nature safety cases are sensitive for a variety of reasons (security, confidentiality, sensitivities) and not many are available in the pubic domain. Some anonymised cases are available from Adelard and John Knight maintains a list of some public cases at [Bishop 90]. There is also use briefing material at [Adelard] on safety and [Repository] on assurance cases. There are also some safety case templates available for UK defence projects.

Goal-based software safety cases have seen take up and interest shown from other sectors. In 1998 the UK CAA Safety Regulation Group drafted a goal based approach to the regulation of air traffic management systems and its proposals are contained in CAP670 SW01, *"Requirements for Software Safety Assurance in Safety Related ATS Equipment"* [CAA 98]. This has gone through a number of iterations and was last issued in 2003. Proposals from Eurocontrol [ESARR] incorporate similar top level goals to CAP670 SW01 and there is a guidance document from Eurocontrol on safety cases along with some examples and an introduction to GSN [Eurocontrol].

The idea of a case has also applied in areas outside the safety arena. In the medical domain there is considerable work on trust cases [Gorski 04] for IT systems and there is an International Working Group on Assurance Cases (for Security) [Bloomfield et al 06] In terms of security, the uptake by the US DHS of Assurance Cases is significant [Repository] as is their sponsorship of the draft international standard ISO/IEC 50126 . The whole issue of evidence based approaches is receiving considerable international interest as indicated by the US NAS study [Jackson et al 07].

There is also work in validating simulation by the use of "cases" – that is whether one can trust the results of a simulation for a new system. This is led by SE Validation, a small UK company.

11.3. Current practice

11.3.1. Goal structured safety cases

We defined a safety case [Bloomfield et al 98] as "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment". More recent definitions (e.g. in revised Def Stan 00-56) make explicit the concept of structured argumentation "A structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment"

Current safety case practice makes use of the basic approach that can be related to the approach developed by Toulmin [Toulmin 58] where claims are supported by evidence and a "warrant" that links the evidence to the claim. There are variants of this basic approach that present the claim structure graphically such as Goal Structuring Notation (GSN) [Kelly and Weaver 04] or Claims-Argument-Evidence (CAE) [Bloomfield et al 98]). GSN is the dominant approach in the UK defence sector. These notations can be supported by tools [McDermid 94][Emmet and Cleland 02] that can help to create and modify the claim structure and also assist in the tracking of evidence status, propagation of changes through the case, and handling of automatic links to other requirements and management tools. However the actual claim decomposition and structuring is normally very informal and argumentation is seldom explicit. In practice, the emphasis is on communication and knowledge management of the case, with little guidance on what claim or claim decomposition should be performed.

Toulmin's scheme addresses all types of reasoning whether scientific, legal, aesthetic or management. The CAE style is much more like Toulmin's where we articulate and elaborate textually and yet retain the overall structure. The philosophical approach is that context and assumptions are often rich and complicated and best captured in narrative. A purely graphical rendering would be simplistic and verbose and would certainly go against the spirit of Toulmin in that reasoning can rarely be reduced to just a flow chart or logic network.

The "case" and associated supporting tools can be seen as having two main roles:

Reasoning and argumentation as an over-arching argumentation framework that allows us to reason as formally as necessary about all the claims being made. Here there are two very

different viewpoints: the one that sees argumentation as primarily a narrative and the other where we seek to model judgements in a formal framework. There are some hybrid approaches where the case can be seen to integrate and communicate a selection of formal analyses and evidence, e.g. it would not seek to reason formally about the timing of a component but leave that to a separate analysis. The balance between these two approaches should be part of on-going research.

Negotiation, communication, trust as a boundary objective between the different stakeholders who have to agree (or not) the claims being made about the system. To this end it has to be detailed and rigorous enough to effectively communicate the case and allow challenges and the subsequent deepening of the case.[8]

One approach used in Adelard can be explained in terms of a "triangle" of:

- The use of accepted standards and guidelines.
- Justification via a set of claims/goals about the system's safety behaviour.
- An investigation of known potential vulnerabilities of the system.

This is presented as a triangle as in Figure 11.1

Figure 11.1: The Safety Case triangle

The first approach is based on demonstrating compliance to a known safety standard—which is the approach that is normally used There are tools available such as the nuclear industry Emphasis tool [Smith and Stockham 07] that supports an initial assessment of compliance with IEC61508.

The second approach is goal-based—where specific safety goals for the systems are supported by arguments and evidence at progressively more detailed levels. The final approach is a

[8] An engineering analogy is to see the case as part of a "signal processing system" – the licensing and certification process – that seeks to reject false claims and accept good ones. For critical systems the probability of false positives has to be very low (i.e., there must be a very low chance of accepting a flawed system).

vulnerability-based argument, where it is demonstrated that potential vulnerabilities within a system do not constitute a problem. This is essentially a "bottom-up" approach as opposed to the "top-down" approach used in goal-based methods.

These approaches are not mutually exclusive, and a combination can be used to support a safety justification, especially where the system consists of both off-the-shelf (OTS) components and application-specific elements. While there is considerable experience in other sectors with the claims-argument-evidence (CAE) structure for safety justification, this type of structuring is novel to the nuclear industry. Feast therefore proposed to use the CAE method as a support for the assessment to provide flexibility without losing rigour, but not as the primary basis.

Adelard has also been developing a more rigorous approach to claim decomposition. While the details have yet to be published this has involved an empirical study of a large number of safety cases available to Adelard (and reflecting the take up of the cases approach) allowed an informal empirical study of what is needed for a claims decomposition language.

The key technical concept behind the work is the idea that claims decomposition can be formalised in a very general way that, while being formal, it allows for rigorous checklist and possible formal application. Normally the proofs would not be able to be discharged without an appeal to a corresponding formal model of the system being considered (e.g. to show that timing is additive).

The claim decompositions identified are:

Main types – keywords	*Comment*
architecture	splitting a component into several others
functional	
set of attributes	splitting a property into several others
infinite set	inductive partitioning (e.g., over time)
complete	capturing the full set of values for risks, requirements, etc.
monotonic	the new system only improves on the old system
concretion	making informal statements less vague

Table 11.1: The Adelard Fog claim decompositions

The soundness of the approach is being established by an encoding in Prototype Verification System.

11.3.2. Confidence, challenge and meta-cases

The structured safety case, either in CAE or GSN notations, need to be challenged and assessed if we are to be sure that it is fit for purpose. In some areas such as defence and nuclear there is a well defined process for such independent assessment.

The basic measure of efficacy of an argument in this work is the *confidence* that the argument engenders in a dependability claim. Informally here, a dependability case[9] is taken to be some *reasoning*, based upon *assumptions* and *evidence*, allowing certain *confidence* to be placed in a dependability *claim*. For a given claim, the confidence – and its complement, *doubt* – will depend upon confidence/doubt in the truth of assumptions, in correctness of reasoning, and in 'strength' of evidence.

A key notion here is the recognition that there is uncertainty involved in the assessment of system dependability: it is (almost) never possible to claim with certainty that a dependability claim is true. In the jargon this kind of uncertainty is called *epistemic* [Littlewood and Wright 07]. It concerns uncertainty in an expert's 'beliefs-about-the-world'. It contrasts with the more common *aleatory* uncertainty, which deals with 'uncertainty-in-the-world': e.g. uncertainty about when a software-based system will fail next. It is now widely accepted – even for software-based systems – that the latter is best measured using probabilities, such as probability of failure on demand.

In this work, probability is used to capture the epistemic uncertainty involved in the dependability case: it is the confidence in the truth of the dependability claim. An important part of the work investigates how effective multi-legged arguments are in increasing such confidence (i.e. over and above the confidence that would arise from one of the legs alone). This work has been published in the open literature: see [Bloomfield and Littlewood 03] [Littlewood and Wright 07].

The results from this work concerning the efficacy of multi-legged arguments are, at first glance, not surprising. For example, it is shown that:

- There are benefits from the use of multi-legged arguments, compared with the single legs (the work only treats 2-legged arguments so far);
- These benefits fall short of what could be expected if arguments 'failed' independently (e.g. if you have two argument legs, for each of which you obtain 10% doubt in the

[9] This is usually a *safety* case, but the ideas apply more generally to other dependability attributes such as reliability and security.

dependability claim, then you cannot expect your doubt to fall to 1% when they form the legs of a '1-out-of-2' argument).

But the work is more interesting than these bald results suggest, in two ways. Firstly, the formal probabilistic treatment of confidence in claims is novel. It treats rigorously what is often ignored, or treated very informally, even in safety critical standards. It could be used to satisfy the recommendation arising from the SOCS report [HMSO 86] that an ACARP (As Confident As Reasonably Practical) principle be introduced into safety cases. So far, however, only theoretical modelling work has been done and its practicality needs to be proved on real safety cases, or realistic case studies. There also needs to be further work on how the formalism might fit into current regulatory practice: see [Bloomfield and Littlewood 07].

Secondly, the detailed study of idealised safety cases, as in [Littlewood and Wright 07], demonstrates how there can be subtle and non-intuitive interactions between the different – and usually disparate – components of a safety case. Although this example concerns a multi-legged case, the insights apply to any safety case in which confidence in a dependability claim rests upon disparate evidence (and this is, essentially, always). The BBN methodology used in this work – which retains a complete analytical description of the uncertainty – seems much more powerful than the usual numerical BBN treatments.

This work is not yet in a state where it could be taken up as the basis for tools to be used to help build safety cases. More work is needed in several areas – e.g. on the difficult problems of eliciting probabilistic beliefs from experts for input to the Bayesian analyses. On the other hand, it has given some novel insights and it points the way toward more rigorous ways of constructing quantitative probabilistic safety cases.

11.3.3. Other research

In addition to the work cited above, there has been a variety of other research into safety cases: work on modularity within the avionics sector [Adelard], work sponsored by HSE on assessing Software of Uncertain Provenance (SOUP) [Jones et al 01][Bishop et al 02a][Bloomfield and Littlewood 07][Bishop et al 02b][HMSO 86] and US work on fallibility and other issues [Greenwell et al 06]. A large amount of research has been sponsored by the nuclear industry, particularly in the UK.

Within the European nuclear industry, the Cemsis project – Cost Effective Modernisation of Systems Important to Safety (2001-4) – sought to *maximise safety* and *minimise costs* by developing common approaches within the EU to the development and approval of C&I

systems that are regarded as "systems important to safety" (SIS) that use modern commercial technology. The project had close contacts with the Task Force on Licensing Safety Critical Software of the Nuclear Regulator Working Group (NRWG) of the EU DG for Energy and Transport. The main results of the project are guidance documents on a proposed formal approach to safety justification of SIS [Courtois 01], on requirements engineering for SIS and a qualification strategy for 'Commercial Off-The Shelf' (COTS) or 'pre-existing' software products. These were evaluated in a number of industrial-based case studies including a 'public domain' example that was used to explain and illustrate the guidance. The approach to new build in the UK specifically distinguishes claims, argument and evidence in the licensing requirements for the Generic Design Assessment for new reactors.

11.3.4. Specific tool support

Tool support for safety cases can be considered in three broad categories:

Decision support and elicitation tools. These allow one to expose the thinking behind the argument, advise on how to construct a case, and assist in reading and review. They are considered in more detail below.

Tools to generate evidence. These provide the evidence that support the safety case argument. They include safety analysis tools (fault trees, FMECAs), tools for collecting and analysing field experience, static analysis , test and proof tools.

Safety Management System Infrastructure support. In this category there are the tools for configuration management and traceability such as Requirements Engineering support tools and Hazard Logs.

The most commonly deployed tool specific to graphical safety cases is the Adelard ASCE tool. There is considerable research and development of alterative types of tool and integration with different environments.

There is currently standardisation effort with the OMG on claims-argument-evidence and this should provide a good foundation for interoperability of tools and longevity of case documentation.

11.3.5. Current research issues

The description of the current state of the art identified research on claim decomposition, structuring of cases and confidence and illustrates the close coupling between practice and research in this area. Such an agenda should include:

Extending and consolidating the application of the approach.

Development of cases for specific classes of system for less critical and ultra critical systems. (e.g. using probability of correctness (as opposed to reliability) as in [Littlewood 00]. The relationship between cases and the variety of sector standards and generic standards such as IEC 61508

The generalisation of the concept to security and other attributes (not as straightforward as it might seem) and the incorporation of threat assumptions

The broadening of cases to socio-technical systems

The use of cases in adaptive systems (e.g. real-time cases)

The ability to deal with COTS and systems with uncertain provenance (SOUP)

Developing approaches to structure and justify the structure of cases (without defeating its communication role)

A more rigorous method for claim decomposition

A rigorous approach to compositionality of cases

Understanding the role of diversity in argumentation

Understanding and exploiting the relationship between the argument structure and the architecture of a system

Communication

The interdisciplinary assessment of cases as a means of communication between stakeholders.

Confidence and challenge

The modelling and evaluation of confidence in cases and developing an approach to meta-cases. The propagation of confidence in cases and special arguments styles and approaches.

An interdisciplinary approach to stopping rules and claim limits – technical, psychological, and sociological

Evidence

The integration of different types of evidence: operational experience; formal proof of certain properties, process evidence

The relationship between a case and a formal proof of correctness of a system property: more generally the relationship of a case to machine supported reasoning.

The role and limits of statistical testing.

11.4. Concluding remarks

The start of the art of safety case for computer based systems has to be addressed within the context of regulation and system level approaches to safety. A structured approach to safety cases for computer based systems has been developed that addresses both the reasoning that safety properties are satisfied as well as providing an effective approach to communicating this reasoning. The acceptance of a case is (or should be), in the end, a social process.

The use of goal-based, structured cases is very appealing, supporting as it does innovation and flexibility but as can be seen from this review much work is needed to develop a case and put it on a convincing footing. While the basis of Toulmin's scheme is really very simple the industrialisation and application to complex systems is a significant undertaking. Our current approaches rely very heavily on the expertise and best practice of the community and the challenge and review that cases receive. The work is normally not published as there are sensitivities in most real cases and even the research that is being done is not well represented in the literature. This paper has attempted to identify public domain sources of information for those interested in the field.

12. Online Monitoring

This chapter surveys forms of online data collection that are in current use (as well as being the subject of research to adapt them to changing technology and demands), and can be used as inputs to assessment of dependability and resilience, although they are not primarily meant for this use.

12.1. Introduction

Monitoring the components of a system can be used to make decisions about the management of the system and thus control its behaviour. Monitoring is also used for debugging and for evaluating in real time the performance or quality of service (QoS) of a system. Monitoring in IT systems can be defined as the process of dynamic collection, interpretation and presentation of information concerning objects or software processes under scrutiny [Mansouri-Samani and Sloman 1992]. This paper provides a general description of monitoring: "*the behaviour of the system is observed and monitoring information is gathered; this information is used to make management decisions and perform the appropriate control actions on the system*". When setting up a monitoring system, technical problems have to be solved in relation to detecting the events of interest, labelling these events with sufficient information for use in measurement, transmitting the event notices to where they will be used, filtering and classifying the events according to the measures of interest, and acting upon the collected information. Online monitoring has long been recognised as a viable means for dependability improvement [Schroeder 1995]. This report surveys the use of monitoring in diverse areas of: dependability, performance enhancement, correctness checking, security, debugging and testing, performance evaluation and control.

In general, when we deal with the monitoring of computing systems we have one or more *monitored elements* (e.g., computers, operating systems, application packages) send out *signals* or *measurements* of their well-being or distress to a *monitoring site* (logically centralised, even when computing elements are distributed), which thus can extract (or help humans to extract) various forms of processed information and make (or help humans to make) decisions. Concrete examples include:

- A computer manufacturer receiving *health/failure data* at a centralised maintenance centre from customer installations of its machines.
- An independent company that installs and maintains machines at client sites polling/receiving *health data* from them.

- A software vendor/distributor receiving reports of failures from its installed software.

Monitoring may serve either or both of the following two purposes:

- Supporting direct feedback control: the monitoring data trigger decisions about, for example, dispatching maintenance technicians; or switching in stand-by components; or switching to a higher-security configuration of a system; or asking developers to investigate a suspected software bug; etc. That is, monitoring helps to improve the monitored system so as to make it more resilient/dependable.
- Providing data for assessing the dependability or resilience of the monitored system (which is the focus of this report).

The rest of this chapter is organised as follows. First, we provide a brief timeline description of monitoring activities. Subsequently, the "Research in Distributed Systems Monitoring" section outlines important issues in *monitoring distributed systems*. In the following sections, we group *forms of online monitoring* according roughly to areas of use, which determine communities of users and researchers. For each such area, we describe pertinent industrial and academic work, focusing on its potential for dependability assessment. These "areas" are not intended as a partition of all related work, based on first principles, but just as a convenient clustering of existing practices, research and literature into (possibly overlapping) categories.

12.1.1. A concise historical perspective of monitoring

The field of monitoring and related telemetry activities for dependability assurance can be regarded to be as old as the electronic engineering discipline.

In the early 20th century the design and implementation of dependable systems have not been rigorously constrained by cost and schedule limitations as is the case nowadays - *"In many cases, high levels of reliability were achieved as a result of over-design"* [Smith 2001]. Quantified dependability assessment was, therefore, largely non-existent.

The maintenance procedures for early days' computers consisted of a technician inspecting the state of the vacuum tubes. *"Computers built in the late 1950s offered twelve-hour mean time to failure. The vacuum tube and relay components of these computers were the major source of failures; they had lifetimes of a few months."* [Gray and Siewiorek 1991]. These tasks were daunting, and in the case of a large number of simultaneously faulty vacuum tubes almost impossible, since the locations and causes of the errors were difficult to determine. The reliability and ease of maintenance have been significantly improved by using various tools, such as cathode-ray oscilloscope [Chandler 1983], as in the case of Colossus, the

famous World War II codebreaker machine that had to provide a 24-hour service. Signal voltages were examined with an oscilloscope and it was used to assess the effect of relative timings of different parts of a circuit.

The post-war age hastened the advent of intricate mass-produced component parts, which brought with them greater complexity and parameter variability. This resulted in poor field reliability of the assembly-line products. As a result, for example, the focus of military equipment in the 1950s shifted to the collection of field failure data of safety-critical systems and the interpretation of the corresponding test data. Databases holding failure rate information were created by such organisations as UKAEA (united Kingdom Atomic Energy Associations), RRE (Royal Radar Establishment), RADC (Rome Air Development Corporation, USA), [Smith 2001].

An ability to detect, log and react to dependability-relevant events has been part of the design of, at least, high-end dependable computers since the 1970s and 1980s. During these decades, several computer vendors (IBM, Tandem, Digital Equipment Corporation (DEC), etc.) used electronic communication (usually a secure telephone link) to maintain and service their customers' computer devices. As a result, data for maintenance as well as for statistical studies of field dependability were made available. For instance, IBM offered customers of its 3090 high-end mainframe series (which appeared around 1985) an option for automatic remote support capability (an automatic call was initiated to IBM) or a field service engineer was sent to the customer's site after a maintenance job has been raised due to a *processor controller*[10] detecting errors (see [Siewiorek and Swarz 1998, Chap. "General-purpose Computing"]). All this information was logged in a central database called RETAIN, and, for example, a service engineer could communicate with it to obtain the latest maintenance information. Another option offered with the 3090 series was that a field technical support specialist could establish a data link to a processor controller to remotely monitor and/or control the mainframe. A similar option existed as a part of the well-known VAX system from DEC (VAX-11/780, the first member of the VAX computer family, was introduced in 1977). The user mode diagnostic software of the VAX system reported errors, which initiated calls to the DEC's Remote Diagnostic Centre (RDC). Subsequently, an RDC engineer would establish a data link to the customer's VAX system through a special remote port and execute

[10] Processor controller is a unit of, usually mainframe, computers, which handles the functions of maintenance, monitoring, and control.

diagnostic scripts, logging everything during the process (see [Siewiorek and Swarz 1998, Chap. "General-purpose computing"]).

Other examples of mature computer-based technology that employed forms of online monitoring for achieving high dependability include:

- Telephone switching systems from the pioneer in the field, AT&T Inc. For example, *"the 3B20D processor diagnostic capabilities detect faults efficiently, provide consistent test results, protect memory content, allow automatic trouble location and are easy to maintain"*, see [Siewiorek and Swarz 1998, Chap. "High-Availability Systems"]. These systems employ dynamic redundancy (e.g., resuming a computation on another processor once an error is discovered) as the means for achieving high availability. Error detection techniques are numerous. The high availability processors of the 3B series (3B20D/3B20C/3B21D/3B21E) were first produced in the second half of the 1970s. The 4ESS switch, the first digital electronic toll switch for long distance switching (introduced in 1976), initially used the 3B20D processor. It is worth noting that 4ESS employed advanced error reporting techniques. The switch had automated procedures in place for analysing error data, with the purpose of correlating and identifying intermittent errors which evaded standard diagnostic mechanisms.
- Long-life unmanned spacecrafts, like deep-space planetary probes *Voyager* and *Galileo*, which use block redundancy for fault tolerance. They are capable of performing error detection, diagnosis and reconfiguration automatically (on the spacecraft) or remote control (from ground stations), see [Siewiorek and Swarz 1998, Chap. "Long-life systems"] for more details.

Following the use of, most commonly, secure telephone links for collecting customers' failure reports in the 1970s and 1980s, a natural step forward was to automate the collection of failure data: failure data of computers/applications were automatically collected and sent back to the manufacturer/vendor, after what the data were forwarded to the engineering department for analysis. Consequently, much research on dependability has been based on data from automated error logging ([Iyer 1995] lists many early examples), demonstrating how measurement can inform modelling to lead to assessment (prediction) of dependability in future operation.

The technological advancement for reliability improvement has been noticeable in the hard disk drives industry in the 1990s, when IBM introduced Self-Monitoring, Analysis, and

Reporting Technology (S.M.A.R.T.) and the closely related Predictive Failure Analysis (PFA) technology. Dependability monitoring mechanisms are now often packaged in off-the-shelf components, e.g., hard disk drives. Nowadays, many hard disks are equipped with S.M.A.R.T., which detects and reports various reliability indicators and administrators can use it for maintenance and potentially for reliability assessment. As computer systems became increasingly distributed and integrated with networks, this kind of monitoring has taken on more varied forms.

12.2. Research in Distributed Systems Monitoring

In this section we provide a general introduction to the problem of *monitoring* and *on-line management* of distributed systems, with an emphasis on academic research.

Distributed systems are more difficult to design, build and monitor, than centralised systems because of i) parallelism among processors, ii) random and non-negligible communication delays, iii) partial failures (failures of components of the system) and iv) lack of global time base and strict synchronisation between distributed nodes. The main issues of monitoring distributed systems are as follows:

- *No central point of decision making*: the process of making decisions in a distributed system may itself be distributed (leading to, for example, the "agreement problem").
- *Incomplete observability*: in some cases it is not possible to observe certain parts of the system, resulting in incomplete (and possibly inconsistent) information about events in the system. Thus usually, locally observed events are collected and some entity, using information about these events, has to build a global view of the system (an example of data fusion problem).
- *Non-determinism*: distributed, asynchronous systems are inherently non-deterministic. Thus, two executions of the same program may produce different, but correct, orderings of events. This makes the reproduction of errors and the re-creation of test conditions difficult.
- *Vast data volumes*: The huge amounts of data produced by monitoring have to be processed (possibly in real-time, so as to perform short-term management, or to filter the logs to avoid the difficulty of storing the large amounts of data).

Somewhat old, but still useful overviews of research problems in monitoring distributed systems can be found in [Joyce, Lomow et al. 1987] and [Mansouri-Samani and Sloman

1992]. Important research results in the field of distributed systems, which affect the problem of monitoring them, include:

- Seminal research related to the identification of the global state of a distributed system: work on logical clocks and vector clocks [Lamport 1978]; clock synchronization [Lamport 1987], [Cristian 1989]; global snapshots [Mani-Chandy and Lamport 1985] and a comparison between the concepts of group membership and system diagnosis [Hiltunen 1995]. These results are important in the field since the authors identified problems and solutions needed for the correct monitoring of distributed systems.
- Results about "Unreliable failure detectors" [Chandra and Toueg 1996], dealing with the problem of monitoring crash failures of components in distributed systems.
- Quality of Service (QoS) monitoring and management: challenges and approaches in providing QoS monitoring are described in detail in [Jiang, Tham et al. 2000]; the idea of a "dependability manager" (a special type of QoS monitor) was defined in [Porcarelli, Bondavalli et al. 2002] and specialized in [Porcarelli, Castaldi et al. 2004].
- The whole field of extracting information from an event log (possibly presented as a real-time data stream) is relevant to the needs of monitoring for dependability assessment. Among recent developments, the work on adaptable parsing of real-time data stream [Campanile, Cilardo et al. 2007] proposes methods and tools to support monitoring of a e.g., data stream under real-time constraints.
- Relationships between monitoring and metrology (see [van Moorsel 2009, Chap. "Metrology"]): recent work about monitoring distributed systems [Falai 2007] defines a conceptual framework for experimental evaluation and monitoring activities that supports rigorous (from a metrological point of view) observation of distributed system.

12.3. Automatic Failure Reporting for Software

Commercial and open-source software increasingly employs automatic failure reporting features.

A discussion of these monitoring mechanisms [Murphy 2004] lists benefits for both the vendors and the customers; vendors are enabled to extend their quality assurance processes, while customers are offered improved user experience (the user's perception of the product's quality, often seen as the most important quality measurement, can be continuously improved). This paper however warns that certain pre-requisites are necessary during

development of a failure reporting system: i) an understanding of the customer base and usage profile, ii) an understanding of the failure profile, and iii) the users' privacy, frequently regarded as the most sensitive requirement, has to be respected.

For software development organisations, the main role of automatic failure reporting is usually detecting the bugs that manifest themselves most frequently (e.g., in [Murphy 2004] it is claimed that Windows XP exhibited very skewed failure set immediately following its release – "a very small number of bugs were responsible for the majority of customer failures"), thus triggering and prioritising the bug fixes. However, these data could also be used for assessing reliability of the software during use, if only data about the amount of use were also collected. Reliability will of course vary between users, depending on their "operational profiles" (frequencies of different kinds of demands on the software). So, collecting simple usage data, like total time in operation, or number or volume of demands on an installation, would be enough for retrospectively assessing the reliability of that installation, but extrapolation to reliability in future use requires linking failure reports to frequencies of demand types, or other characterisations of operational profiles. Extending failure reporting with automated collection of accurate usage data and operational profiles was proposed by Voas and colleagues [Voas 2000] to help vendors make better long-term decisions with respect to their customer requirements (benefits to vendors would include, for example, detecting misused and unused features or discovering the most typical machine configurations used to execute the vendor's software). A scheme for collecting operational profile data, related to the work in [Voas 2000], is one of the claims in the US patent 6862696 [Voas 2008].

The other important limitation of automated failure reporting systems as usually deployed is that they typically focus on detecting crash failures, as this is an easy requirement to satisfy, but ignore "value domain" failures (production of wrong results), which however should be considered at least as important. The costs of the two kinds of failures vary between applications, but the importance of non-crash failures should not be underestimated: for instance, bug reports for SQL servers show a surprisingly high prevalence of defects that cause non-crash failures [Gashi, Popov et al. 2007], [Vandiver, Balakrishnan et al. 2007].

[Garzia, Khambatt et al. 2007] present an approach for assessing end-user reliability. In essence, the approach is based on the following:

- Grouping users' computers in terms of the number of disruptions they experience per unit of time. Three groups are defined: *excellent*, *good* and *poor* reliability group. The

grouping is performed in regard to the anticipated improvement of the Windows Vista over Windows XP SP2 operating system.

- Placing each user's machine in one of the three groups based on the frequency of failures on a particular build, over a period of two weeks (for this the data from the installations of beta users were used).
- Identifying bugs and assessing reliability were then employed for every major Windows Vista milestone.

The results of the assessment approach are used for ensuring the reliability objectives at major software release stages have been met. The authors focus on end-users of desktop and laptop computers; these users seek disruption-free operation of the software – both a software failure and a planned activity, e.g., installation of a software update, is regarded as detrimental. The case study was performed with Microsoft Windows Vista operating system and, according to the paper authors, the outcomes of the proposed approach have been used in deciding on the product's shipment strategy.

The insightful work of [Li, Ni et al. 2008] provides a classification of current approaches used in the IT industry for collection of reliability feedback data – four classes are identified: interactive user reporting, online user reporting, automated per-incidence reporting and automated reliability monitoring. The classification, though possibly incomplete, together with seven proposed evaluation criteria, helps comparing different approaches. The authors recommend that automated reliability monitoring is used as the approach for reliability assessment of mass-market software, and they give details of Windows Reliability Analysis Component (RAC), an implementation of the recommended approach. RAC is available on Windows Vista operating system; it runs as a low-priority process; it collects the data using operating system event logs and system calls and periodically sends the information to Microsoft. The information includes details of all detected failures during an observation period, and it provides the basis for data normalisation by reporting successful executions and execution durations.

The limitations of the automated reliability monitoring approach can be identified reflecting on, for instance, the above cited article [Li, Ni et al. 2008]:

- The authors acknowledge that software producers could be interested in reliability feedback data for two reasons: evaluation of the reliability status of the product, and identification and reaction to the failures. Their viewpoint, however, focuses on the latter,

whereby the collection of reliability feedback data is primarily undertaken for software improvement. This viewpoint might be inadequate in some cases, e.g., during the software procurement phase when only assessment results are needed. The stance of the software producers is, however, expected, for rarely is a software vendor willing to share assessment results (if they exist at all) with the party in charge of procurement.

- Crash failures are assumed (as is the case with most automatic failure reporting systems, see above) - the focus of the study is on application crashes, application hangs and operating systems crashes.
- Even if the fault-model is to include the faults leading to non-crash failures, the *correctness*, one of the seven evaluation criteria proposed in the paper, is defined with regard to a "perfect" oracle – it is based on the assumption that the software specifications are correct. Chances are, however, that due to developers' misunderstandings of users' needs, the users perceive the software to malfunction (observing a non-crash failure for example) even if it is running correctly according to the developers' specification. This may lead to non-diagnosable failures.

We outline below two of the established software failure reporting systems.

The Mozilla Foundation has been providing automatic bug tracking through a proprietary system called *Talkback*, or *Quality Feedback Agent* (QFA), since the 1990s. The strategy was inherited from its parent company Netscape and had been successful in that, in the 2000–2004 period, the QFA program helped to resolve around 778 bugs [McLaughlin 2004]. Mozilla report [McLaughlin 2004] that in their experience a user base of 20,000 or more provide the best information for the purpose of uncovering software faults, building reproducible test cases and producing appropriate fixes; we may note that confidence in dependability estimates also increases with a larger number of monitored installations. Beginning with Firefox version 3 alpha 5, the Mozilla platform is equipped with an open-source crash reporting system, which is a combination of the following products:

- Google Breakpad [Google 2008a] client and server libraries.
- Mozilla-specific crash reporting user interface and bootstrap code.
- Socorro Collection and reporting server [Google 2008b].

Before the introduction of the abovementioned RAC [Li, Ni et al. 2008] feature in the Vista operating system, Microsoft™ offered their customers automated per-incident reporting

through *Windows Error Reporting (WER)* [Microsoft 2008]. This framework provides for extraction of crash dump data, which are forwarded to Microsoft for further analysis. Whereas crash, hang, and kernel fault reporting is provided by default in the Windows Vista operating system, application-specific issues have to be communicated through WER – the client-side WER service forwards the memory dump information from a desktop application, which had crashed or stopped responding, to Microsoft. This action is initiated only after the user has given his/her consent. Reported problems trigger a mix of automated and human-mediated reactions like:

- A solution is sent to the user.
- More information is requested from the user so to aid developers in resolving the problem.
- A new "case" for the problem is opened, and the user is notified by a status message.

There are other software products that provide automated, opt-in systems for failure reporting, such as The Apple Crash Reporter [Apple 2006], the GNOME [GNOME 2008] and KDE [KDE 2008] desktop environments. Apart from these integrated solutions, there exist more comprehensive, stand-alone products which enable one to collect, quantify, and control large amounts of test and field data, such as FRACAS by Relex™ [Relex™ 2008].

Academic research has followed the pace of advance in automatic failure reporting. To take an example, *Cooperative Bug Isolation* [Liblit 2004] is a low-overhead instrumentation strategy for collecting information from a multitude of software end-users. The focus of the work is a *statistical debugging* approach. It provides a suite of algorithms for finding and fixing software errors based on statistical analysis of feedback data collected from the end-users.

Many interesting ideas about (automated) failure data analysis were presented at the "Reliability Analysis of System Failure Data" workshop held at Microsoft Cambridge, UK on 1-2 March 2007. For instance, one suggestion [Fetzer 2007] was that execution traces to be replayed at the developer's site following a failure, be collected through a "hybrid" approach: instead of executing automatic tracing constantly and incurring (unnecessary) overhead during normal operations, one could combine it with the RX approach [Qin, Tucek et al. 2005] for software fault tolerance. The RX approach provides a light-weight checkpointing mechanism, which ensures that a program is rolled back to the most recent checkpoint after an error is discovered. During the retry of the program, the environment is changed to maximize

the probability that bugs are masked. A possibility, then, is to initiate collection of execution traces only during the retries, which would reduce the overhead during normal operation.

Online monitoring plays an integral role in the field of *online failure prediction*. This field's main concern is the assessment of the risk that "misbehaviors" result in failures in a near future. For this purpose, online failure prediction employs runtime monitoring and measurement of the current system state. This kind of monitoring enables detection of errors through observation of side-effects on the system, e.g., a memory leak can be discovered through exceptional circumstances like high CPU load or disk activity, or an atypical function call. A comprehensive survey of online failure prediction techniques is given in [Salfner, Lenk et al. In print].

12.4. Network Monitoring

Network monitoring is a part of network management functions. It includes the following: means for identification of problems caused by network failures and overloaded and/or failed devices, continuous measurement of QoS attributes to enable corrective management actions or to produce longer term dependability measurements, as well as means of alerting network administrators to virus or malware infections, questionable user activity and power outages. Network monitoring systems differ from intrusion detection systems (IDSs) or intrusion prevention systems (IPSs)[11] in that their focus is not exclusively on security, but also network functioning during ordinary operation. In addition to being used for availability assurance and performance improvement, network monitoring helps to accumulate information that could be used for planning future network growth.

Network monitoring systems typically use "network monitoring interface cards" (NMIC), pieces of hardware similar to standard NICs, but with a difference in design so that they passively listen on a network. NMICs have no MAC (Media Access Control) layer addresses and are invisible to other devices on the network. One of the most prominent manufactures is Endace Ltd. (http://www.endace.com/dag-network-monitoring-cards.html).

Network monitoring systems may listen to different application level protocols like FTP, SMTP, POP3, IMAP, DNS, SSH, TELNET. As for communicating data, many protocols can be used, but the standard is Simple Network Management Protocol (SNMP), defined by the Internet Engineering Task Force (IETF) in their Requests for Comments, RFC 1157 [IETF 1990] and RFC 3413 [IETF 2002]. The protocol prescribes an architecture in which a

[11] Both IDSs and IPSs are discussed in the section "Intrusion Detection and Intrusion Prevention Systems" of this report.

software component (denoted as *agent*) executes on managed systems (network elements such as hosts, gateways, terminal servers, and the like), collecting the desired network information, that is in turn communicated to one or more managing systems. It exists in three versions; SNMPv3 has been the standard version since 2004. The IETF has designated SNMPv3 a full Internet Standard, the highest maturity level for an RFC. An interesting extension of SNMP with accounting features is given in [Stancev and Gavrilovska 2001].

Closely related to SNMP is the RMON standard for remote network monitoring, which prescribes the ways of monitoring the internet traffic. It has been standardized by RFC 1271 in November 1991, but it is updated by RFC 1757 [IETF 1995] in February 1995. RMON2 [IETF 1997] is an extension of RMON that focuses on higher layers of traffic above the medium access-control (MAC) layer.

Network monitoring software products are either embedded in larger software suites or are offered as stand-alone products. A comprehensive repository of network monitoring tools can be found at [StanfordLinearAccelerationCentre 2008].

Numerous QoS monitoring tools have been built by industry. QoS monitoring is made more difficult by the fact that various pieces of network equipment route traffic through numerous firewalls and address translators. Cisco™, for instance, offers several tools for QoS monitoring [Cisco 2008]. Another example of QoS monitoring tool is XenMon [Gupta, Gardner et al. 2005], a performance monitoring tool based on Xen virtual environment, [Xen 2008]. XenMon provides support for resource allocation and management functions through accurate monitoring and performance profiling infrastructure.

12.5. Intrusion Detection and Intrusion Prevention Systems

Whereas network monitoring is primarily concerned with checking and reporting from "inside" a network, e.g., detailing measurements of link bandwidth, latency and response from routers and switches, and CPU utilisation time, *intrusion detection systems* (IDS) monitor a network with the aim of discovering threats coming from *outside*, to trigger appropriate defensive reactions.

IDSs are an important mechanism in the assessment of network security. They are used to collect data about attacks (see, for example, [Cukier et al. 2006], [Leurre.com 2008], [HoneynetProject 2008]), an important part of security research. Detecting both attempted and successful intrusions would of course allow measurements of security (measurements of both absolute frequency of successful intrusions and of the fraction of intrusions that are

successful), but this potential is limited by the fact that the IDSs themselves are inevitably imperfect, and exhibit *false negatives* (failing to raise an alarm following a real intrusion), *false positives* (raising an alarm due to legitimate traffic) or *event misclassification* (raising an alarm following a real intrusion, but failing to classify it correctly).

IDSs are classified into different categories, which bring different imperfections in their suitability as measurement instruments, on the basis of various criteria, such as:

- The *location* of the monitoring sensors:
    - Network intrusion detection systems (NIDS) identify intrusions by examining network traffic, e.g., Snort [Snort.org 2008].
    - Host-based intrusion detection system (HIDS) identify intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, etc.) and other host activities and state, e.g., OSSEC [OSSEC 2008].
- The *monitoring focus*:
    - Protocol-based intrusion detection systems (PIDS) monitor and analyze the traffic of a protocol.
    - Application protocol-based intrusion detection system (APIDS) works similarly to PIDS, but the focus is an application layer protocol.
- The *detection method*:
    - Anomaly-based, e.g., Cfengine [Cfengine 2008], which can be configured to perform anomaly detection, i.e. to detect data patterns that differ from the normal, established behavioural patterns.
    - Signature-based, network traffic or host activity is analysed so that predetermined attack patterns, known as signatures, are discovered.

Expectedly, there exist hybrid intrusion detection systems, which combine two or more approaches, e.g., *Prelude* [Prelude 2008] or *Emerald* [SRI 2008].

An alternative classification groups IDSs into *passive* and *reactive* [Tipton and Krause 2003]. The latter are also known as Intrusion Prevention Systems (IPS). In addition to standard detection, logging and signalling of intrusions, they disable the connections initiated by the suspected malicious sources.

A list of intrusion detection systems is given in [TimberlineTech 2008], and a classification in a tree-like structure can be found at [IPSec 2008]. Comprehensive guidance on intrusion detection is given in the NIST's standard [Scarfone and Mell 2007]. A solution for online flow-level intrusion detection for high-speed network, denoted as HiFIND, is given in [Gao, Li et al. 2006]. HiFIND's major characteristics are: it is scalable with respect to flow-level detection on high-speed networks; it is resilient to DoS (Denial of Service) attacks; it can distinguish SYN flooding[12] and various port scans for effective mitigation; it enables aggregate detection over multiple routers/gateways; and separates anomalies to limit false positives. The approach for resilience improvement of intrusion detection systems in [Sousa, Bessani et al. 2007] combines proactive and reactive recovery. The complementary approach maintains system's correct operation by ensuring the availability of minimum amount of replicas.

Yet another way of detecting intrusions is given in [Strunk, Goodson et al. 2002]. The authors propose a new technology for storage devices to safeguard data, even when the host operating system is compromised. The technology enhances the intrusion survivability through:

- Faster intrusion detection through monitoring of (suspicious) storage activity.
- Facilitating intrusion diagnosis by providing a wealth of data to administrators.
- Simplifying and speeding up post-intrusion recovery by: preserving all pre-intrusion states, providing low granularity restoration (any number of files can be restored) and enabling users to continue utilising the system (non-destructive recovery).

12.6. Web Services Monitoring

Web services technology represents a significant support element for the growing integration of ICT systems across organisational boundaries. The World Wide Web Consortium (W3C) adopts the following broad definition: "*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network*", [W3C 2008].

Web services have generated increased interest in QoS monitoring. Web Service Level Agreements (WSLA) are contractual agreements between two parties, either customers and service provider(s) or service providers only, which formally specify the Quality of Service

[12] A SYN flooding is a form of DoS attack in which an attacker keeps sending SYN (synchronise) messages to a target's system, until the target's system resources are exhausted.

(QoS) level to be provided, as well as the consequences resulting from its violation. Contractual obligations are commonly stipulated between provider and consumer of a service, as well as the financial consequences in case the obligations are violated. A language specification for WSLAs is given in [Ludwig, Keller et al. 2003]. WSLA monitoring and enforcement is increasingly important; this is particularly true for environments where dynamic or on-demand subscriptions to web services are possible. The monitoring is important for customers, who need the reliability and performance agreements to be maintained at runtime, and indeed to service providers who want to control the agreed level of usage. The unpredictable Internet environment implies that reliability guarantees of the applications using web services become difficult to ensure: this increases the importance of web service reliability assessment techniques. Research has been active in the web service monitoring field. Monitoring web services should include measuring at multiple sites; both client- and server-side measurements should be taken. To this end, the authors of [Sahai, Machiraju et al. 2002] propose an automated and distributed SLA monitoring engine, the Web Service Management Agent (WSMN), which provides for joining and correlating web service and business process data. WSMN is based on the proposed specification language that allows for definition of accurate and flexible WSLAs. A framework for specifying and monitoring WSLAs is given in [Alexander and Heiko 2003][13]. It integrates flexible and extensible formal language (based on XML) for expressing service level agreements and a runtime architecture comprising several SLA monitoring services. The framework is flexible in that it can be applied to not only SLAs of web services, but to other inter-domain management scenarios, such as management of networks, systems or applications. In [Raimondi, Skene et al. 2008] WSLA contractual conditions are represented as timed automata and a methodology, with a prototype implementation using an Eclipse[14] plug-in, for WSLA online monitoring is proposed.

A possible architecture, and a corresponding implementation, for performance management of web services is given in [Levy, Nagarajarao et al. 2003]. The feedback controller of the management system, which tunes the performance parameters (e.g., load balancing or request queuing), is based on a continuous monitoring approach, wherein system components (servers, gateways, global resource manager, and console) share monitoring information through a publish/subscribe network (e.g., the global resource manager computes the

[13] Further information can be found on the webpage http://www.research.ibm.com/wsla/.
[14] Eclipse, http://www.eclipse.org/, is at the time of writing recognised as one of the most popular software development platforms.

maximum number of concurrent requests that a server should execute, based on the performance information *published* by each gateway).

Besides the contractual aspect of QoS monitoring, there is a need for users to assess or rank the dependability of web services that they might use. Chen et al. propose WSsDAT tool for dependability assessment of Web Services from the client perspective by collecting metadata representing a number of dependability metrics [Chen, Li et al. 2006]. One of the purposes is to identify good replication strategies, which employ the same or similar web services to enhance reliability. WS-DREAM [Zheng and Lyu 2008] addresses these issues by allowing users, potentially geographically dispersed, to carry out web service reliability assessment in a collaborative manner. This worldwide testing, and sharing of test case results, is coordinated by a centralised server; reliability assessment becomes affordable to many participants through data sharing. What both WS-DREAM and WSsDAT seem to be missing in assessing dependability levels is the account of the impact of operational profile from each individual client.

12.7. Embedded Systems Telemetry

The use of *embedded systems* has been continuously growing for decades. They have been integrated in portable devices such as mobile phones as well as in automotive safety systems, e.g., anti-lock braking system (ABS), but also in nuclear power plant control and safety systems.

The trend has required advances in monitoring of these systems for supporting hardware and software maintenance, as well as dependability and performance assessment. As manufacturers tend to embed "telemetry" capabilities to monitor the performance and health of the hardware which the computers control (from engines in aircraft, trucks and cars to factory equipment and house appliances) and support maintenance and repair, so they can add similar capabilities to monitor the performance and health of the embedded computers. Like other monitoring practices, these can be used to evaluate the dependability and resilience of these systems.

A sort of online monitoring is frequently employed for error detection, e.g., as in "watchdog" technique, where a small coprocessor is used to monitor the behaviour of the main processor. An insightful survey on this technique is given in [Mahmood and McCluskey 1988]. This technique is not exclusive to embedded systems but is also used for dependability assessment of general-purpose computers.

Monitoring embedded systems as a part of runtime verification process requires non-intrusive arrangements, e.g., with hardware resources dedicated to the monitored system. These concerns are addressed in [Watterson and Heffernan 2007]. A number of existing runtime verification tools are referenced, highlighting their requirements for monitoring solutions (see the section "Runtime Verification" of this report for more details on online monitoring for runtime verification). The authors present a review of established and emerging approaches for the monitoring of embedded software execution.

A few examples may illustrate current practice in telemetry of embedded systems. Micro Digital Inc™ (founded in 1975), one of the first companies to offer embedded-systems software, recommends the use of Web Network Management Protocol (WNMP)[15]. They claim the following features:

"*WNMP (Web Network Management Protocol) enables simple and flexible administration and monitoring of embedded systems using ordinary web browsers. In contrast to SNMP, no special program running on the client system is needed. Nor is a MIB (Management Information Base)[16] or MIB tool required. This makes it easy to administer or monitor an embedded system from any computer, even a modern cell phone*" [MicroDigital 2008].

The software tool *µC/Probe*, from *Micriμm* [Micrium 2008], enables developers to monitor embedded systems in a live environment. Execution traces are obtained without the need for halting the application, unlike conventional debuggers, and thus the benefits extend beyond the development process. This type of monitoring application is particularly valuable for embedded systems that cannot be suspended to examine variable and program flow, e.g. compressors used to pump natural gas through a pipeline, or similar reciprocating engines (like car pistons). Bringing an engine online is a very complex task and the variables have to be examined at runtime to validate the working of the control system. On the other hand, such an approach carries an overhead (with respect to performance and resource usage) due to additional computation needed for communication between the *µC/Probe* and the target embedded application. Such an overhead is a characteristic of other monitoring-related features, which generally can be provided at additional cost if spare resources are available.

[15] http://barracudaserver.com/brochures/WNMP.html

[16] *"A management information base (MIB) stems from the OSI/ISO Network management model and is a type of database used to manage the devices in a communications network. It comprises a collection of objects in a (virtual) database used to manage entities (such as routers and switches) in a network."*, http://en.wikipedia.org/wiki/Management_information_base

Advanced Telemetry Linked Acquisition System (ATLAS) [McLarenElectronicSystems 2008] is a software package from McLaren Electronics used for obtaining, displaying and analysing data from automotive control systems. It has capabilities of comparing live telemetry data with uploaded logs. Also, it provides fast data handling so that the large quantities of data could be processed in real-time.

The research led by professor Brian Williams from Computer Science and Artificial Intelligence Laboratory (CSAIL), at Massachusetts Institute of Technology, has led to a significant progress in the field of *model-based autonomy* [Williams and Nayak 1996], [Williams, Ingham et al. 2003]. This research advocates the use of the model-based programming paradigm for addressing intrinsically difficult task of programming complex embedded systems, which necessitates reasoning about complex interactions between sensors, actuators, and control processors. Statistical models, such as hidden Markov model, are constituent parts of model-based programming. This type of programming has led to the creation of *Livingstone*, a core component in the flight software of Deep Space One, the first spacecraft in NASA's New Millennium programme. Livingstone is a reactive, model-based and self-configuring kernel for autonomous systems. One of the main active research areas in the field is concerned with intelligent embedded systems, which are able to explore, diagnose and repair themselves using online assessment combined with adaptation. The online assessment approach relies on the advanced monitoring capabilities.

12.8. Monitoring Large-Scale Enterprise Software

Business integration across enterprise domains requires connection of data and applications throughout the enterprise. In this section, we discuss monitoring techniques for large, enterprise-wide, systems of networks and computers. Nowadays, enterprise software systems represent the main support for many businesses to provide services to the end users. The software is required to provide continuous service despite failures of individual hardware components or software processes.

Satisfying stringent dependability requirements is, however, hard – enterprise software is growing in size and complexity, and the necessary updates are becoming more frequent [Munawar and Ward 2006]. Each component in a system requires monitoring of its resources, performance and state variables. This frequently leads to generating an overwhelming amount of data, which are hard to collect and store, and are complicated to analyse. State-of-the-art monitoring methods, therefore, focus on continuously examining only partial data, intensifying monitoring level only when a particular problem is suspected [Munawar and

Ward 2006]. This improves performance by decreasing measurement effects and at the same time reduces storage, transmission, analysis and reporting overhead.

There is a variety of commercial solutions for enterprise software monitoring, such as:

- The IBM's MQ WebSphere [Stanford-Clark 2002] family of products (formerly know as MQ Series) offers a solution. The products transform field data into whatever form is required by different applications and a publish/subscribe model enables the receipt of the exact subset of data required. MQIsdp (MQ Integrator, SCADA device protocol) offers a publish/subscribe model over TCP/IP with different levels of delivery assurance: at most once, at least once and once-and-once-only.
- Windows Management Instrumentation (WMI) [Microsoft 2000] defines a non-proprietary set of environment-independent specifications which allow management information to be shared between management applications. WMI prescribes enterprise management standards and related technologies that work with existing management standards, such as Desktop Management Interface (DMI) and SNMP (see section "Network Monitoring" of this report). Distributed Management Task Force (DMTF 2008) (DMTF, formerly "Desktop Management Task Force") is a standards organization that develops and maintains standards for management of enterprise IT and Internet environments. DMTF standards, among others, include the following:
    - *Common Information Model (CIM)* - The CIM schema is a conceptual schema that defines how the managed elements in an IT environment (for instance computers or storage area networks) are represented as a common set of objects and relationships between them. CIM is extensible to allow product specific extensions to the common definition of these managed elements. CIM is the basis for most of the other DMTF standards.
    - *Web-Based Enterprise Management (WBEM)*, a set of systems management technologies, which specifies the following: protocols for the interaction between systems management infrastructure components implementing CIM, a concept for defining the behaviour of the elements in the CIM schema, the CIM Query Language (CQL) and other specifications needed for the interoperability of CIM infrastructure.

Many products from leading software vendors are WMI enabled, such as:

- HP OpenView, a former Hewlett Packard product range consisting of network and systems management products. In 2007, HP OpenView was renamed to HP Software [HewlettPackard 2008]. HP OpenView is most commonly described as a suite of software applications which allow large-scale system and network management of an organization's IT infrastructure.
- IBM Tivoli Management Framework (TMF) [IBM 2008b]. TMF is a systems management platform from IBM – the framework is a CORBA-based architecture that provides management of large numbers of remote locations or devices.

IBM's *autonomic computing* initiative [IBM 2008a], of which IBM Tivoli is one technology, has addressed the growing complexity of computing systems management through self-managing autonomic system paradigm. The approach, whose concept originates from social animal collective behaviour, defines four functional areas: self-configuration, self-healing, self-optimisation, and self-protection. Online monitoring is pertinent to self-healing and self-protection, but also to self-optimisation, since the area is concerned with control of resources through automatic monitoring to ensure the optimal functioning with respect to the defined requirements. Academic research has provided significant contributions to the autonomic computing programme, and at the same time to monitoring of enterprise level software. Some of the notable examples are:

- Astrolabe [van Renesse, Birman et al. 2003], a DNS-like distributed management service for scalable management and self-organization of large-scale, highly dynamic, distributed applications.
- A mechanism for dynamically monitoring and reconfiguring a system-of-systems built out of a heterogeneous mix of components [Kaiser, Gross et al. 2002].

Research in the field of dependability of enterprise systems still suffers from scarcity of empirical data, with few published empirical studies [Sahoo, Squillante et al. 2004] [Hsueh 2008]. The earlier paper [Sahoo, Squillante et al. 2004] emphasised the need for a deeper understanding about the occurrence of failures and their statistical properties in real (enterprise) environments. The authors presented a detailed empirical and statistical analysis of system errors and failures from a production environment of 400 servers, pointing out that the collection of empirical data is more difficult with the increased complexity of today's software. The latter paper [Hsueh 2008] presented a characterisation of failure data collected

by Boeing's *internal incident and problem recording system*. The characterisation focuses on only service availability related records obtained from this online monitoring system.

12.9. Runtime Verification

Runtime verification techniques are means for runtime failure reporting and they typically rely on "formal methods" for designing the functions that detect failures (and are thus not limited to crash failures). These techniques are motivated more with verifying and improving software than by measuring its dependability attributes. Well-established, heavyweight formal methods, e.g., theorem proving or model-checking, are being increasingly accompanied by online (runtime) monitoring with the aim of helping in verifying program correctness. This is, at least partly, due to the constant increase of software complexity and size (frequently resulting in state space explosion), which limits the use of traditional verification techniques. Moreover, runtime error detection solutions are explicitly required by some development standards of safety critical industries: e.g., EN 50128 [BSI-BritishStandards 2001] for computerized railway control systems prescribes several methods and techniques that are mandatory or highly recommended for achieving specific *safety integrity levels* (SIL).

Monitoring for software failures during software runtime confirms (or disproves) whether an actual run of the software conforms to the requirements specification, by checking whether it preserves certain formally specified properties. Note that runtime error detection is closer to testing than to exhaustive model checking since it cannot prove the *lack of defects*, but is only able to indicate the *presence of defects* in the execution. Runtime error detection techniques are used for detecting defects that have remained in the software even after rigorous testing. Error detection signals emitted by these solutions can trigger error confinement and recovery activities, i.e., runtime verification can be seen as an entry point of fault tolerance mechanism. The actual implementation of runtime verification techniques can be built on such increasingly popular approaches as *aspect-oriented* (AOP [Kiczales, Lamping et al. 1997]) or *monitoring-oriented* programming (MOP [Formal-Systems-Laboratory 2008]).

The seminal paper [Diaz, Juanole et al. 1994], which extends the ideas from [Ayache, Azema et al. 1979], proposes the *Observer* concept for validating runtime behaviours of self-checking distributed systems – "*systems that detect erroneous behaviours as soon as errors act at some observable output level*". The concept is based on the principle of an observer-worker system. A system of this type consists of two distinct components, a *worker* and an *observer*: the former is a standard implementation of the system behaviour; while the latter is a redundant implementation which outputs are comparable with the outputs of the worker. Using the

concept, it is possible that a system's runtime behaviour is (continuously) checked against a formal specification.

StateRover tool [TimeRover 2009] provides, among other features, support for validation, automatic test generation and verification of system requirements. The tool is based on an automatic model checking technique for UML (Unified Modeling Language) statecharts. The self-acting attribute of the technique stems from the use of an automatic white box test-generation combined with automatic run-time monitoring of statechart assertions. The white-box test generator automatically produces test sequences (events, conditions, and input data); chooses one of the test sequences and submits it to the System Under Test (SUT), which subsequently starts up an embedded assertion for run-time monitoring.

A taxonomy of runtime software-fault monitoring tools is described in [Delgado, Quiroz Gates et al. 2004]. It presents common building blocks of a monitoring system: specification language, monitor and event-handler (the results of monitoring are captured and communicated to the user by an event-handler, and in some cases responses to violations are initiated). Operational issues, such as program types targeted by the monitoring system, platform dependencies and tool maturity level, are also included in the taxonomy.

MOP [Formal-Systems-Laboratory 2008] is a tool-supported formal software development framework in which runtime monitoring presents a basic design principle. MOP enables a developer to specify required properties by using a formalism of choice. Many formalisms are allowed, due to the observation that different types of logic are best suited to different types of specifications. Depending on the success of the properties' validation (a property can be violated or successfully validated), a specific code, stated by the programmer, will be executed.

Another interesting work, which builds on the extensively studied field of self-checking via assertions, is the *RunTime Reflection Project* [Arafat, Bauer et al. 2005]. The project establishes methods to dynamically analyze reactive distributed systems at runtime. The approach is layered and modular; it provides the means for both detecting system failures (through runtime verification) and identifying the failure causes (through a detailed diagnosis). Diagnoses can be subsequently used in order to trigger recovery measures, or to store detailed log-files for off-line analysis. The approach has many points in common with the proposal of the Dependability Manager [Porcarelli, Bondavalli et al. 2002]; the main difference is that runtime verification has as its main purpose *verification* of the system, while

the main purpose of the Dependability Manager is the *reconfiguration* of the system based on runtime evaluations.

There is a significant body of research dedicated to the automatic synthesis of executable assertions (i.e., the implementation of checks to be performed during runtime). A significant part of the work appears in the Runtime Verification Workshop series [RVW 2008], such as [Rosu and Havelund 2001], [Finkbeiner and Sipma 2004], [Gerth, Peled et al. 1995].

13. Robustness Testing

Robustness is an attribute of dependability which measures the behaviour of the system under non-standard conditions. Robustness is defined in IEEE Standard 610.12.1990 as the degree to which a system operates correctly in the presence of *exceptional inputs* or *stressful environmental conditions* [IEEE]. To further refine the difference between robustness and fault tolerance Avizienis *et al.* defined robustness as "dependability with respect to external faults, which characterizes a system reaction to a specific class of faults" [Avizienis 04].

The goal of *robustness testing* is to activate those faults (typically design or programming faults) or vulnerabilities in the system that result in incorrect operation, i.e., robustness failure. Robustness testing mostly concentrates on the faults activated through the system interface. The robustness failures are typically classified according to the CRASH criteria [Koopman 00]: **Catastrophic** (the whole system crashes or reboots), **Restart** (the application has to be restarted), **Abort** (the application terminates abnormally), **Silent** (invalid operation is performed without error signal), and **Hindering** (incorrect error code is returned - note that returning a proper error code is considered as robust operation). The measure of robustness can be given as the ratio of test cases that exposes such faults, or, from the system's point of view, as the number of robustness faults exposed by a given test suite.

Robustness testing can be characterized with the following two components of tests (stimuli); the *workload* triggers a regular operation of the system, while the *faultload* contains the exceptional inputs and stressful conditions applied on the system. Depending on how these two loads are balanced, robustness testing can be used for verification and evaluation purposes. Robustness testing can be used as a special kind of conformance testing, where only a faultload is executed against the public interfaces of the system. Submitting only a high amount of workload, i.e., executing a stress test, can also assess the robustness of a system.

Robustness benchmarks are defined by specifying the workload and faultload, the standard procedures and rules to execute them, the experimental setup, and the relevant measures that characterize the robustness of the system under benchmarking. As robustness is an attribute of dependability, robustness benchmarks can be considered as a special case of dependability benchmarks (see Section 5.2).

From the point of view of the lifecycle of systems, on one hand robustness testing can be used as an internal step in the development process, complementing the other verification and

evaluation activities. On the other hand, robustness tests can be performed (typically by external parties) after the release of the system, to assess its dependability or to compare it to other systems. Another typical use of robustness testing is to identify components that need wrapping. This is particularly relevant when the system contains commercial off-the-shelf (COTS) components, which cannot be modified to correct the faults/robustness weaknesses, thus a wrapper built around the component is the only solution to improve its robustness.

The rest of the chapter is organized as follows. The first section gives a historical overview of the different basic approaches used for robustness testing. The second section presents how these techniques are used in different applications domains, and how the approaches have to be adapted to the specialities of the given domain. Finally, the third section concludes the chapter.

13.1. Overview of basic approaches and major supporting tools

In the past decades many research projects were devoted to the robustness testing of a specific application or application type. The early methods were mainly based on low-level hardware fault injection, but later the research focus moved to higher-level software faults and vulnerabilities because they can be more easily reproduced and are closer to typical design or programmer errors. The main milestones can be connected to the introduction of new *testing techniques.*

13.1.1. Injecting physical faults

Early work on robustness testing used fault injection (FI) tools to induce or simulate the effects of various hardware related faults. Here a clear distinction can be made between the purposes of general FI and FI for robustness testing. The general technique assesses the ability of a system or component to handle internal hardware or software faults. In a robustness testing framework, FI can be used to assess the ability of a component to handle interaction faults that are triggered by injecting faults into the environment (e.g., interacting components or underlying layers) while keeping the tested component intact. In this way also the robustness of error detection and error handling mechanisms (considered as components to be tested) can be investigated. To precisely target FI, detailed information about hardware usage (e.g., memory areas) must be available. FIAT [Barton 90] or FTAPE [Tsai 96] are examples for FI tools that are reported to be used for such robustness testing purposes.

13.1.2. Using random inputs

One of the easiest robustness testing techniques is to generate random input for the system. Fuzz [Miller 95] was one of the first of such tools. It was utilized in three series of experiments to test reliability of various applications. In 1990, utility programs on seven variants of Unix operating systems were tested. In 1995, the tests were repeated to check whether robustness of these utilities had been improved and support to test X Window applications were added. Lastly, in 2000, Fuzz was used to test 30 GUI applications on Windows NT. Although the method used was really simple, they managed to catch a great deal of robustness errors, namely 40% of the Unix command line programs and 45% of the Windows NT programs crashed (terminated abnormally) or hung (stopped responding to input within a reasonable length of time) when called with random input data.

Although random testing is a basic technique, it proves to be useful even for modern COTS software systems. The tests in Fuzz were reapplied to MacOS in a study prepared in 2006 with the following results: 10 command line utilities crashed out of the 135 utilities that were tested (a failure rate of 7%), 20 crashed and 2 hung out of the 30 GUI programs tested (a failure rate of 73%). Thus, it turns out that robustness testing using random inputs is still a viable technique as robustness of common software products has not been significantly improved in general in the last fifteen years.

Fuzzing is extensively applied to security related testing, as presented in a recent book [Takanen et al. 2008].

13.1.3. Using invalid inputs

The basic software robustness testing technique is to select the values applied in interface calls from the boundaries and outside of the allowed input domain. E.g., if a parameter accepts a positive integer then robustness tests may contain zero, a negative number and the MAXINT value. A typical example was the Riddle tool [Ghosh 99], which used a grammar-based description of the system's input to generate random and invalid tests. According to the grammar definition legally structured inputs were generated filled with random values, possible malicious values (like special or non-printable characters) and boundary values (for example numbers like MAXINT + 1 or very long inputs). In this way, syntactically correct inputs could also be created and not just totally random streams, and a greater portion of the systems functionality could be accessed by the tests. Results showed that about 10% of the tests on GNU utilities produced an unhandled exception. The robustness failures observed were mainly memory access violation exceptions, privileged instruction exceptions and illegal

instruction exceptions. The possible cause of these exceptions was the improper handling of the non-printable characters in the input and the long input streams.

13.1.4. Introducing type-specific tests

The robustness tests can be refined if specific invalid inputs are used for each function in the system's interface. To minimize the amount of manually created test cases a type-specific approach was recommended. The valid and invalid inputs were defined for the data types used in the system's public functions, and the robustness tests were generated by combining the values for the different parameters. The size of the invalid input domain can be further reduced by using inheritance between the types to test. The Ballista tool [Koopman 00] used this approach as depicted in Figure 13.1 to compare the robustness of 15 POSIX operating systems using a test suite for 233 function calls. The goal of the research was to implement methods to measure the robustness of the exception handling mechanism of systems. The results could be used to evaluate the dependability of the system and characterize how it responds to the failures of other components. In an experiment performed on the Safe Fast I/O (SFIO) library, the performance drawback of robustness hardening was also measured. The tests showed that the performance penalty of proper validation and parameter checking was fewer than 2%.

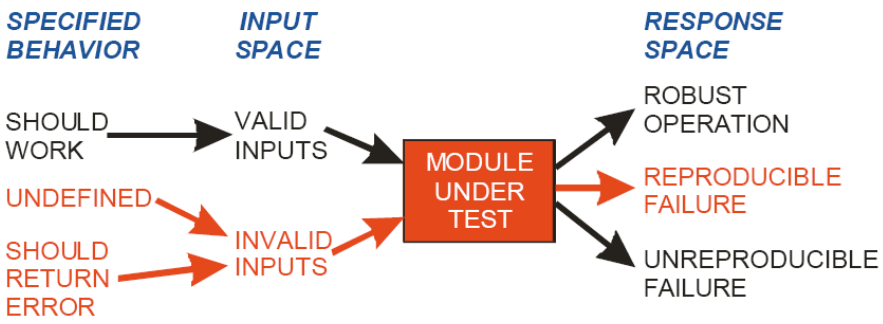


Figure 13.1: Robustness testing approach [Koopman 00]

13.1.5. Testing object-oriented systems

The above type-specific technique can be enhanced in object-oriented systems with the help of automatically building a parameter graph with the type structure. The JCrasher tool [Csallner 04] creates robustness tests for Java programs automatically by analyzing which methods return a type needed for the actual parameters. It examines the type information of a set of Java classes and constructs code fragments that will create instances of different types to test the behaviour of public methods with random data.

13.1.6. Applying mutation techniques

Code mutation techniques [DeMillo 88] can be also applied to generate robustness tests. Starting from a valid code, e.g., a functional test or an application using the system's interfaces, mutation operators are applied, which resemble the typical faults causing robustness problems (e.g., omitting calls, interchanging calls, replacing normal values in parameters with invalid values).

13.1.7. Penetration testing

A penetration test is a proactive and authorized attempt to compromise information security [Arkin 05]. In other words, penetration testing is a type of robustness testing applied for security assessment. The process involves an active analysis of the system for any potential vulnerabilities that may result from poor or improper system configuration, known and/or unknown hardware or software flaws, or operational weaknesses in process or technical countermeasures. It plays an important role to characterize systems deployed in a hostile environment (e.g., networked systems). Black/white/grey box techniques are distinguished depending on the knowledge about the internal implementation details, simulating in this way attackers with different familiarity with the system. Robustness evaluation and penetration testing are interrelated: on one hand, robustness evaluation of networked systems may include the evaluation of resistance to intrusion, on the other hand, the testing of classical robustness problems forms an important aspect of penetration testing as robustness problems may open a way for intrusions or data leakage.

13.1.8. Historical overview of the basic approaches

A detailed survey of the above techniques was provided in the context of ReSIST NoE, in particular in the report summarizing the state of knowledge [ReSIST 06b].

To conclude the section on basic approaches and major supporting tools, Table 13.1 presents the historical time line of the techniques and evaluates their current applicability, while Figure 13.2 presents the relations between the techniques.

Table 13.1. Historical time line of techniques used for robustness testing

Testing technique	First introduced	Evaluation of applicability
Injection of physical faults	Early 1990s	Used in specific domains, e.g., in safety-critical systems
Random inputs	Mid 1990s	Basic technique, still useful for current software

Invalid inputs	Late 1990s	Not used on its own (part of type-specific testing)
Type-specific testing	Around 2000	Very effective technique, complemented by mutation
OO approach	Early 2000s	Extension of type-specific tests to OO languages
Mutation techniques	Early 2000s	Complements type-specific techniques
Penetration testing	Early 2000s	Focused on security-related faults

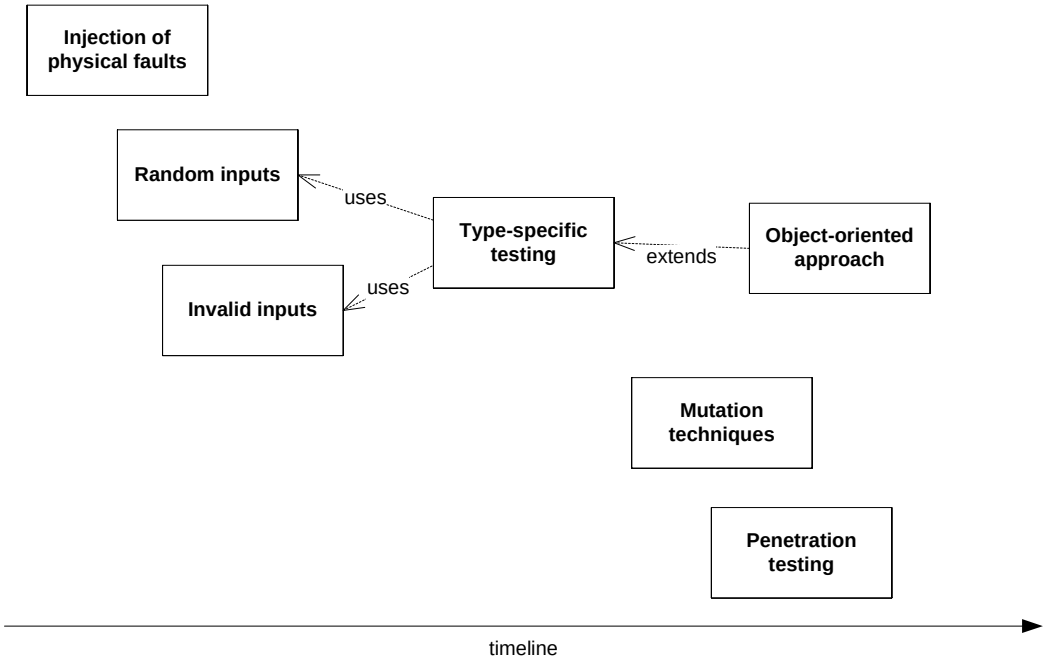


Figure 13.2: Relations between the basic robustness testing techniques

13.2. Techniques in specific application domains

In the recent years, robustness testing techniques have been successfully used in several application domains. In the following we survey the peculiarities of these techniques.

13.2.1. User interfaces

State-based testing of graphical user interfaces (GUI) relies on building a graph based model that describes the elements of the interfaces and the connections (e.g., allowed activation sequences) between them. This model is often called an event-flow or event-interaction graph. The graphs describing the normal operation are completed with the connections that are not allowed. The activation of these connections (after reaching their initial states by a sequence

of normal interactions) constitutes the robustness test suite. These robustness tests can be automatically generated based on the possible sequences obtained from the graph model.

The GUITAR framework [Memon 07] is a tool set for creating automatic tests for GUIs. It detects the elements of the GUI, e.g., menus, buttons, and constructs an event flow graph. Rapid tests trying to crash the application can be generated from the model after each release of the software. The effectiveness of the method and the tools were demonstrated on the GUI of an open-source office suite.

13.2.2. High availability middleware

Robustness is a key factor in middleware systems that are applied to provide high availability services to applications and to manage the configuration of redundant components, since robustness failures in the middleware can be activated by poor quality application components, and one such component may render the whole application inaccessible. The complexity in testing these middleware comes from the highly state-based nature of these systems: most of the calls in the public interface need a proper setup code to produce any effect.

In [Micskei 07] the common interface specified by the Service Availability Forum was used to compare the robustness of different middleware implementations. A robustness test suite was generated from the description of the common interface using a type-specific approach. This suite was used to test the system's robustness against invalid inputs. To test stressful environmental conditions an operating system call wrapper was used, hence the manifestation of many stressful conditions (network errors, lack of disk space etc.) can be simulated by injecting errors returned by system calls. The different methods exposed distinct robustness faults in the system, demonstrating the importance of using mixed testing techniques.

13.2.3. Real-time executives

The use of microkernel technology constitutes an attractive technological alternative to developing operating systems for numerous applications. Microkernels are currently a common component on a wide range of applications, from daily-use appliances (e.g., mobile devices) to space-borne vehicles. Robustness testing of microkernels has been addressed by the research and industry community using stressful operational conditions, invalid input at the public interfaces, and using fault injection.

In [Arlat 02] the MAFALDA tool is used to provide objective failure data on microkernels and to improve its error detection capabilities using the Chorus and the Lynx microkernels. A

specialized version of that tool, the MAFALDA-RT, is used in [Rodriguez 02] to address real-time systems. The tool proposes a novel method to cope with the problem of temporal intrusiveness caused by the use of SWIFI tools. In addition to typical failure modes (e.g., application hang, system hang, exception, etc), the observation capabilities of MAFALDA-RT extend to temporal measurements characterizing both the executive and application layers (e.g., task processing, task synchronization, context-switch, system calls, etc.).

The Xception tool [Costa 03] is a SWIFI tool that was used in several works of microkernel robustness testing (e.g., [Maia 05], [Barbosa 07] and Moraes 07]). This tool is able to introduce perturbations at the processor level to emulate errors at the hardware or software level (in the case of software fault, the fault is introduced when user applications are being executed). The robustness evaluation consists of assessing the abilities of the microkernel to handle the stressful situation caused by the existence of faults. One interesting work used Xception to emulate hardware transient faults and observe the effects at the user application and at the operating system level [Madeira 03]. This work showed that errors occurring in one user-mode application can propagate to other user-mode applications through the OS itself.

13.2.4. OLTP and DBMS

Online Transaction Processing (OLTP) systems are not the most common type of systems used in experimental studies of robustness. Operating systems and networked systems studies largely surpass OLTP systems in the available literature. This is a worthy note on future work concerning robustness evaluation research. Although not as common as other types of system, OLTP systems were used in past robustness-related works. The following two are worth noting.

The work presented in [Vieira 03b] shows a dependability benchmark (encompassing the attributes of robustness) of OLTP systems. This work subjected the systems under observation to stressful conditions and measured the performance penalty of the Database Management System (DBMS) engine and the occurrence of data corruption on the data tables. The stressful conditions were caused by the injection of operator errors. This work showed that it is possible to assess the dependability and robustness properties of OLTP systems and rank the systems under study according to the results.

The work in [Fonseca 08] presents a method to detect intrusions and malicious data access based on DBMS auditing. Although this work is not exactly inline with the traditional robustness testing described earlier in this section, assessing the ability to prevent or detect

intrusion can constitute a measure to the robustness of a system (in this case, the security attributes of that system).

13.2.5. Web Servers and Web Services

Web Services are a widespread technology to implement services accessible over a network. Each Web Service has a well-defined interface (usually using a standardized description language). Several methods were proposed to generate robustness tests based on these interfaces for Web Services.

The work described in [Tsai 05] presents specification-based robust testing framework for Web Services. The testing framework includes the analysis of the service specification and ensuring that it is complete and consistent. This work proposes the Covering Scenario Generation algorithm to identify the locations where incompleteness and inconsistency exist. The testing framework also includes the robustness testing of the Web Service by generating positive as well as negative test cases.

The WebSob [Martin 07] is a tool that automates the generation and execution of test cases for Web Services. This tool executes Web Service requests using the provider's Web Service Description Language (WSDL) specification. This tool was applied to freely available implementation of Web Services and revealed robustness problems. The tool does not require the knowledge of the implementation of the Web Services under test.

The work in [Vieira 07] presents a benchmark of the robustness of Web Services using invalid data in the method invocation to discover both programming and design errors. The values used in the method invocations are modified (i.e., corrupted) based on the data types and the meaning of the method parameters. The Web Services are classified according to the failures observed during the tests, which is useful to rank the web services under study.

The WS-FIT tool [Looker 08] performs a dependability analysis of Web Services. It uses the typical methods of robustness testing based on network fault injection. This method includes corrupting, capturing and retransmitting packets. This allows for easy corruption of RPC method invocation within SOAP messages and can emulate the following errors: corruption of input and output data, omission or duplication of messages, delay of messages.

Because web services are exposed to a large number of users through the network, penetration testing plays a relevant part in the robustness characterization of web services. Tests are typically performed using manual or automated tools to systematically break into servers and/or workstations. Currently there are several penetration test suites available, both from

research and industrial initiatives. The following are prominent examples: Core Impact [CST], NeXpose [NeXpose], OWASP WSFuzzer [OWASP], and WSBang [WSBang].

13.3. Concluding remarks

Robustness is a key attribute in many application domains; hence several methods were developed to assess the robustness of a system. They range from physical fault injection and interface testing with invalid input values to sophisticated penetration testing. Sometimes the border between robustness testing and other verification and evaluation techniques is blurred, as the same techniques can be applied in several scenarios.

Although there are mature techniques and tools, further work is needed in several areas. The following challenges can be easily identified:

- Tool-supported *generation of robustness tests* (the advantages of this approach were already demonstrated in some application domains [Micskei 07], but the initial techniques shall be extended to support sophisticated state-based robustness testing techniques as well).
- Integration of robustness testing techniques with the model driven engineering methods (e.g., to specify and configure the test objectives and the related monitoring facilities on the basis of the design models, and to synthesize the necessary wrappers automatically, as demonstrated in [Olah 2009]).
- Elaboration of robustness testing tools and techniques for dynamic (mobile, adaptive, autonomous) systems.

Development of agreed robustness testing processes and test suites on the basis of standardized specifications (like API specifications) in order to support the comparison of various implementations.

14. Human Factors

This document is a brief survey of methods for the quantitative assessment of the effect of the human component on the resilience and dependability of systems including computers. In this document, we will use the word 'system' to refer to the socio-technical system, a system comprising humans interacting with technology.

14.1. Introduction

Human contribution to dependability and resilience has long been recognised by the analysts of systems that include computers (and humans). Humans play varied and increasingly recognised crucial roles in these systems, whether as users or as skilled operators of the computer applications (i.e., what is usually referred to as the 'sharp end' of the system (Hollnagel, 2002)), or as administrators in charge of the maintenance of computerised equipment, or as designers and developers of technology (i.e., roles which are closer to the 'blunt end' of the spectrum).

Humans affect dependability and resilience in multiple ways. Humans can commit errors, for example, because of attention lapses or erratic behaviour in stressful situations. Additionally, dependability and resilience can be jeopardised by malicious human behaviour, for example, in attacks by hackers. On the other hand, people's ability to detect problems and to initiate recovery actions is a positive contribution to the resilience of human-computer systems.

Measurements of human performance in the context of system design and evaluation has been conducted by (cognitive) ergonomics and human factors professionals for many decades, with a strong focus on interface design (Howell, 2008, Moray, 2008, van der Veer, 2008).

Much of the study of human factors in system reliability has focused on human error (Reason, 1999). Human reliability has been a particular concern in the safety of complex engineered systems, in which some ubiquitous presence of computers is just a special aspect of the general issue of human involvement in such systems. Serious incidents, like The Three Mile Island nuclear power plant accident in the late 70s, brought human error to the attention of safety assessment professionals, leading to an increasing interest in measuring human reliability and its contribution to the reliability (or lack thereof) of the whole system. In this context, the historically dominant approach is represented by a series of techniques grouped under the label 'Human Reliability Analysis' or 'Human Reliability Assessment', usually denoted by the acronym HRA. Human reliability is usually defined as the probability that a

person will correctly perform some system-required activity without performing any extraneous activity that can degrade the system. The purpose of HRA methods is to obtain probabilistic information about human error to be incorporated into Probabilistic Reliability Assessment (PRA) or Probabilistic Safety Assessment (PSA). The dominance of these techniques in the last 3 or 4 decades is reflected in the comparatively large space dedicated to them in this survey (Section 14.2). However, we also review here lines of work that go beyond the limitations of scope of conventional HRA methods by obviating some of the shortcomings of these techniques or by considering research questions often overlooked by the HRA community. In particular we review work on: situation awareness and mental workload (section 14.3); empirical studies of automation bias phenomena, i.e., unanticipated effects of computerised "decision support" on human performance (section 14.4); new directions in “resilience engineering” (section 14.5); pointers to studies of human error in simulated scenarios (section14.6).

We present next a brief historical time line of the different approaches to measurement of the human contributions to system dependability and resilience.

14.1.1. Historical Time Line

The following table presents a historical overview of various approaches to measuring human reliability considered in this document. Some of the pointers in the table are spelled out in some more detail in subsequent sections in the document (e.g. in Section 14.2.3).

Period	Historical Highlights
6th c. BCE	Ancient China: early instances of use of concepts related to situation awareness e.g. in Sun Tzu's The Art of War
5th c. BCE	Ancient Greece: documented use of ergonomic principles in the design of tools, jobs, and workplaces
1857 CE	First use of the word "ergonomics" by Polish biologist Wojciech Jastrzębowski (*ergon*=work, *nomos*=natural laws)
Late 19th c. Early 20th c	Frederick Winslow Taylor's "Scientific Management" ergonomic approach, to find the optimum method for carrying out a given task
World War II	Development of new and complex machines: increased focus on human error and cognition, decision-making, attention, situational awareness,…
1950s	Reliability analyses for development of missiles includes human errors
1950s-60s	Use of the term 'situation awareness' in the context of fighter aircrew performance in the Korea and Vietnam wars
1960s	- Swain and colleagues start extensive human error data collection in the nuclear domain - Increasing number of articles related to human reliability and error in journals and conference proceedings
1973	IEEE Transactions on Reliability: Special Issue On Human Reliability
1975	Development of Swaine's THERP, pioneering HRA method, based on 60s data collection
1979	Three Mile Island accident- it highlights serious consequences of (erroneous) human decision making under stress
1980s	Developments in "Cognitive ergonomics": application of cognitive psychology to analyses of human-computer interaction
1981-1983	Further developments of THERP, first manual for this technique published
mid 1980s	A proliferation of HRA techniques (1st generation techniques)
1986	Publication of Dhillon's book: "Human Reliability with Human Factors"
late 1980s	- Validation studies done on many HRA techniques - Development of NUCLARR human error database
1990s	- Birth of 2nd generation techniques - The term 'situation awareness' is adopted by human factors scientists
1990-94	Kirwan's "A guide to practical human reliability assessment" is published
late 1990s-early 2000s	Development of human error databases such as CORE-DATA and HERA
2004	Reliability Engineering and System Safety journal: special issue on HRA
2005-06	NUREG publishes a set of good practices for the implementation of HRA
2008	Halden Reactor Project: International HRA Empirical Study – first results published

14.2. PRA/HRA: Probabilistic Human Reliability Analysis

The Human reliability assessment techniques reviewed here originated as an approach to obtaining human error probabilities that could then be incorporated into Probabilistic Risk Assessment (PRA). The human is seen as yet another component of the human-machine system; in the same way that equipment can fail, so can a human operator commit errors; as such, the human component is susceptible to analyses and measurements similar to those applied to pieces of equipment:. The goal of HRA is thus to assess the risks attributable to human error and to reduce the impact of such error on system reliability.

One major classification of HRA methods distinguishes between first generation techniques (popular in the 1980s) and second generation techniques (developed towards the late 1990s).

Examples of first generation techniques are: THERP (Technique for Human Error Rate Prediction) (Swain and Guttman, 1983), SLIM (Success Likelihood Index Methodology) (Embrey et al., 1984), HCR/ORE (Human Cognitive Reliability/Operator Reliability Experiments) (Hannaman et al., 1985), HEART (Human Error Assessment and Reduction Technique) (Williams, 1986), ASEP (Accident Sequence Evaluation Program) (Swain, 1987).

Examples of second generation techniques are: CREAM (Cognitive Reliability and Error Analysis Method) (Hollnagel, 1998), ATHEANA (A Technique for Human Event Analysis) (USNRC, 2000), JHEDI (Justification of Human Error Data Information) (Kirwan, 1990, Kirwan, 1994) and MERMOS (Le Bot et al., 1999, Meyer et al., 2007).

The discussions that follow will introduce some commonalities and differences amongst the methods. Section 14.2.1 discusses different approaches to quantification in HRA. Section 14.2.2 presents pointers to other dimensions that can be used to characterise and classify these techniques. This is followed by an outline of important developments in the area (14.2.3) and open challenges (14.2.4) in the use of HRA methods.

14.2.1. Quantification in HRA methods

Kirwan (Kirwan, 1994) identifies three main components common to HRA methods: a) Human Error Identification (identifying *what* errors can occur); b) Human Error Quantification (deciding *how likely* the errors are to occur); c) Human Error Reduction (enhancing human reliability by *reducing* this error likelihood). However, as observed by e.g. (Mosleh and Chang, 2004), the level of emphasis on each of these components varies from method to method.

As noted, the original goal of HRA is to obtain a numerical value representing Human Error Probability (HEP), which can then be included in the probabilistic analyses in PRA/PSA. Quantification typically starts by obtaining what is usually called a "nominal" HEP, a basic probability obtained by using established tables derived from human error data collected in simulations (e.g. in THERP), often combined with expert judgement (e.g. in SLIM), and/or informed by cognitive models of human reliability (e.g. in CREAM). A second step is to account for the influence of possible Performance Shaping Factors (PSF) such as task characteristics, aspects of the physical environment, work time characteristics, etc. This influence is expressed as a numerical factor that is used to modify the basic HEP.

In early HRA methods (e.g. THERP), the complex calculations to obtain the final HEP were based on a series of assumptions that later practitioners have come to challenge. One such assumption is that the effects of various performance conditions (PSFs such as interface quality, stress, level of training, task complexity, etc.) do not interfere with each other, thus their effects on human error can be incorporated into the assessment independently.

A reaction to these weaknesses was a larger emphasis on qualitative/descriptive aspects of human performance (i.e. the error identification component of HRA) in second generation methodologies such as CREAM or ATHEANA. These techniques, however, do incorporate guidelines or strategies for quantifying human performance, though in a less prescriptive and reductive way than earlier methods, and putting more emphasis on complex and cognitive (non observable) aspects of human reliability (Reer, 2008a, Reer, 2008b) as opposed to the more behaviouristic approach of earlier techniques.

Studies of HRA have long recognized the aversion of many analysts to quantifying human performance. As highlighted in e.g. (Wreathall, 1981, Reason, 1999), human performance is often perceived as much more complex than that of computer components or as substantially different from automated tools and, as such, not susceptible to the "reductionism" implicit in studying a human as a mere "component" in the system. As a consequence, some approaches to human reliability assessment focus on error identification and assessment, without introducing quantification. We outline next some well known non-quantitative methods used for analysing human reliability.

HAZOP (Hazard and Operability Analysis) (ICI, 1974) and FMEA (Failure Modes and Effects Analysis) (McDermott et al., 1996) are examples of non-quantitative techniques, which although not originally created for assessing human error have been used for human reliability analyses. For example, HAZOP was originally developed for chemical processes

but later adapted to many other systems and operations, including human errors (Leathley, 1997). Similarly, FMEA, originally introduced by the US Armed Forces in the 1940s for military usage, has been later developed for use in human reliability analyses in application domains such as healthcare (Lyons et al., 2004).

Some descriptive techniques have been developed specifically for the analysis of human reliability. For example, THEA (Technique for Human Error Assessment) (Fields et al., 1997, Pocock et al., 2001) is a non-quantitative technique based on Norman's models of human information processing for the evaluation of user interfaces, mainly his error-based three-stage theory of human action, with its goal formation stage, its execution stage and its evaluation stage (Norman, 1988). HEART was developed to help designers of human-computer interactive systems (at the early stages of system design) to anticipate interaction failures or human errors that may be problematic once their designs become operational. Another non-quantitative method, which shares a similar theoretical background to THEA is CHLOE (Miguel and Wright, 2003), a technique for the human error analysis of collaborative work, including both 'social collaborative' (Human-Human) and 'technology-mediated collaborative' (Human-Computer-Human) work. CHLOE uses the results of the analyses to suggest design improvements to support collaborative work.

Obviously, numerical and descriptive approaches are not necessarily incompatible, as exemplified by work by Smith and Harrison (Smith and Harrison, 2002), which complemented results from a numeric method (HEART) with results from a descriptive, scenario based, analysis using THEA. Inevitably, any quantitative model can only be built on the basis of a description of the system. Similarly, Maxion and Reeder (Maxion and Reeder, 2005) report an evaluation of security-sensitive user-interfaces where descriptive analyses based on the THEA technique (as well as Hierarchical Task Analysis (HTA) (Kirwan, 1994)) were combined with the collection of quantitative data of human performance in experimental laboratory settings.

14.2.2. Other Classification Criteria for HRA Techniques

In addition to their different approaches to quantification, HRA techniques differ along many other dimensions, of which we focus here on the following:

a) *Holistic vs. analytic/atomistic techniques.* One could distinguish between "atomistic methods", which conceive events and their causes as entities decomposable into smaller entities that can then be individually quantified ('clear box' approach), and "holistic

methods" where the analysis centres on the entire event, which is typically quantified as an indivisible whole ('black box' approach) (Boring et al., 2005). The atomistic approach is typical of many first generation HRA methods, such as THERP, SLIM-MAUD and ASEP. The holistic approach is characteristic of more recent methods like ATHEANA and CREAM. (Boring et al., 2005) notes that each approach has both advantages and disadvantages and argues that distinguishing between these two approaches allows the analyst to tailor his/her estimation approach to best meet the needs of the event under investigation.

b) *Focus on errors of omission vs. focus on errors of commission*. Most early HRA methods, notably THERP, focused on human errors of omission, that is, those situations where the human user fails to do the action s/he is expected to perform. Later developments have increasingly given importance to errors of commission, that is, actions by the operator that s/he is not expected to perform and lead to undesirable consequences. Errors of commission are obviously more difficult to categorise, analyse and measure, presenting important research challenges for the HRA community (Strater, 2004, Boring et al., 2005, Reer, 2008a, Reer, 2008b).

c) *Focus on human error vs. focus on context of human actions*. This distinction is partly related to the previous two. Traditionally, HRA methods have used individual human actions (individual 'human errors') as a meaningful unit of analysis. However, recent work in second generation HRA methods, in particular CREAM, consider "human error" an ill-defined concept and shift the focus of analysis (informally, the "blame") from individual *human actions* (Johnson et al., 2007) to the *context* (the working conditions) propitiating adverse human actions (Dekker, 2003, Fujita and Hollnagel, 2004, Hollnagel, 2005). This is an important distinction which may have particular implications, for example, for the analysis of collaborative work (work by a team and not just an individual). It is also important to note that the whole debate on HRA has a quasi-political dimension in that some authors see focus on human error quantification as linked to "blame culture" and other organisational attitudes seen as unhelpful to safety.

d) *Application domain*. As noted by (Reason, 1999) developments in HRA have been tightly linked to developments in the nuclear power industry. Early methods like THERP and more recent techniques like JHEDI and ATHEANA were developed specifically with the nuclear power context in mind, although some have been later adapted to be used in other domains, including healthcare applications (Lyons et al., 2004). Some other techniques

have been developed for different application domains, for example: TESEO (Tecnica Empirica Stima Errori Operatori or "Empirical technique to estimate operators' errors") (Bello and Colombari, 1980) was developed in a petrochemical industry context; TRACEr (Technique for the Retrospective and predictive Analysis of Cognitive Errors) (Shorrock and Kirwan, 2002) and CARA (Kirwan and Gibson, 2007) were specifically developed for, and applied to, air traffic management, s, etc. Finally, other techniques (such as HEART) were developed as generic methods not tied up to any specific application domain.

14.2.3. Important Developments in HRA

Brief, up-to-date historical introductions to HRA methods can be found in (Kirwan et al., 2008, Reer, 2008a, Reer, 2008b). Reason (Reason, 1999) provides pointers to reviews and evaluations of HRA in the 1980s; earlier accounts can be found in (Siegel et al., 1975). Here we focus on some important developments.

Human reliability analyses can be traced back as far as the 50s in military applications (e.g. reliability in the use of missiles) and the seminal work that was to be the basis for THERP, one of the earliest HRA techniques, started in the 60s with Swain and colleagues' extensive data collection in the nuclear power domain. The manual for THERP was published in the early 80s (Swain and Guttman, 1983). This is probably the most used, most influential, best studied and most criticised HRA method (Reason, 1999).

The mid 80s saw, in fact, a proliferation of techniques, developed largely in response to the Three Mile Island accident in 1979. From here the approach became commonplace in the nuclear industry, and spread to industries such as oil and gas, and the chemical industry. Well known examples of techniques developed in this period include SLIM, HCR/ORE, HEART, ASEP. Most of these are variations of THERP or applications of essentially the same approach but to different domains. First validations of such techniques took place in the late 80s, as summarized in (Kirwan, 1996, Kirwan, 1997). These evaluations highlighted the inadequacy of some early techniques, for example, HCR/ORE and SLIM (Kirwan et al., 2008).

Various researchers have noted general limitations of first generation methods (Dougherty, 1990, Swain, 1990, Parry, 1995, Hollnagel, 1998, Reason, 1999, Mosleh and Chang, 2004, Sheridan, 2008). Shortcomings of first generation techniques include:

– simplistic assumptions of independence (e.g. between performance shaping factors)

- almost exclusive focus on errors of omission
- difficulty of obtaining precise probability estimates (especially for rare unexpected events)
- need for expensive large scale simulation experiments if one wants to apply the methods rigorously
- excessive reliance on analysts' judgments and risks of inter-judge variability
- lack of causal explanations (or models) of operator error
- limited attention to human ability to recover from error (i.e., positive human contributions to resilience)
- focus on the 'sharp end' of the socio-technical system (i.e., the operators), as opposed to the 'blunt end' (managers, designers, policy makers), which is increasingly viewed as equally (or even more) relevant

Concerns for these limitations and simplistic assumptions led to the development of second generation HRA techniques in the 90s. Earlier sections have already indicated some differences between these newer techniques and first second generation HRA. Other characteristics of second generation techniques include:

- the use of a theory-based error taxonomy (predominantly guided by psychological cognitive models) in contrast with the simple error taxonomies used by first generation methods (which typically only distinguish between omission and commission errors, with a focus on the former)
- a move towards a graded (more sophisticated) match of error scenarios to error identification and quantification (in contrast with a more simplistic "fits"/"doesn't fit" dichotomy in first generation methods)
- these methods are more contextual to the problem/industry to which they are applied and tend to be more domain specific than first generation methods

Two of the most commonly studied second generation techniques are CREAM and ATHEANA. CREAM is a based on Hollnagel's cognitive model of human performance, known as Contextual Control Model (COCOM) (Hollnagel, 1993), with its emphasis on the role of control and contextual factors. ATHEANA (USNRC, 2000), on the other hand, was presented as a method for representing complex accident reports within a standardised structure, using a robust psychological framework.

New developments in the area, including the emergence of second generation methods, were echoed, in 2004, in a special issue on HRA in the journal *Reliability Engineering and System Safety*, which focused on 'data issues' (i.e. how to get appropriate data for quantification of human contribution to reliability) and 'errors of commission' (in an attempt to cover a wide spectrum of potential impacts of human actions on safety) (Strater, 2004).

Most second generation techniques are still under development but some of them have already been included in recently published evaluations and comparison studies, such as (Dang et al., 2008a, Dang et al., 2008b, Forester et al., 2008, Kirwan et al., 2008, Reer, 2008a, Reer, 2008b). No general standards as such exist for the use of HRA in PRA/PSA. Important instances of guidance documents include IEEE's guide for incorporating human action reliability analysis for nuclear power generating stations (IEEE-1082, 1997) and, more recently, NUREG-1792, a report published by the U.S. Nuclear Regulatory Commission, which proposes a set of good practices for the implementation of HRA (Kolaczkowski et al., 2005). This work was followed by NUREG-1842 (Forester et al., 2006), which compared a collection of HRA techniques commonly used in the U.S. against those good practices. In this comparison, the report took into account dimensions such as scope, underlying model, underlying data, quantification approach, strengths and limitations. Among other outcomes, the NUREG-1842 report concludes: 'the various methods have different strengths and limitations, but can be viewed together as a "tool box" that provides different capabilities, some of which are better suited than others for various applications. Having such a tool box allows HRA analysts to effectively address, considering the current state-of-the-art in HRA, the range of human actions (…) needed to be addressed in PRAs and many risk informed applications' (NUREG-184, p. xxxiii (Forester et al., 2006)).

In parallel to these developments, from the late 1980s onwards, there has been an increasing interest in building up human error databases, which is essential to inform numerically the assessments to obtain meaningful predictions. Examples of this development are: the NUCLARR database (Gertman et al., 1988), CORE-DATA (computerised operator reliability and error database) (Kirwan et al., 1997, Basra and Kirwan, 1998) and NRC's Human Event Repository and Analysis (HERA) (Hallbert et al., 2004, Hallbert and Kolaczkowski, 2007).

14.2.4. Open Challenges

We highlight below two important challenges in the analysis, assessment and measurement of the human component:

- *Need of more data of human performance* (especially in naturalistic scenarios). In spite of advances in the collection of empirical data of human performance and the creation of human error databases and repositories (Gertman et al., 1988, Hallbert et al., 2004, Kirwan et al., 2008), a great deal of work is still needed in this area. Quantification in HRA still very much relies on the elicitation of expert judgments and estimates, with the resulting inevitable subjectivity and limited reproducibility of the assessments and quality of the predictions. Another related issue is the lack of established standards for the use of HRA techniques (or measurement of the human component, in general) when incorporated into PRA/PSA.
- *Complexity and scalability of the analyses*. Progress has been made recently (especially in the so called second generation HRA techniques) in considering increasingly complex representations of human errors/actions and the influencing contextual factors. However, most HRA techniques focus on the single-user computer interaction scenario, whereas human-computer interaction increasingly takes place in complex socio-technical environments and most work is actually performed in groups/teams within complex networks, often requiring remote interactions. We have already mentioned the CHLOE technique (Miguel and Wright, 2003), which looks at collaborative work, but this is essentially a descriptive, non-quantitative method. More work is needed, within the HRA community (and the human-computer interaction community at large), to collect and analyze data about human performance in such complex scenarios.

14.3. Studies of Situation Awareness & Mental Workload

A important contributor to human error in human-machine systems is known as situation awareness or SA (Endsley, 1995, Endsley et al., 2003, Banbury and Tremblay, 2004). It is defined as "the perception of elements in the environment, the comprehension of their meaning, and the projection of their status in the near future" (Endsley, 1995) or, more informally, "knowing what is going on so you can figure out what to do".

As reported in (Endsley et al., 2003), in at least one study, 88% of human error was found to be due to problems with SA.

The term SA, can be traced back to World War I, where it was recognized as a crucial component for crews in military aircraft. But it was not until the 1990s that it was widely adopted by human factors professionals, firstly in aviation and military applications but eventually also to such diverse areas as air traffic control, nuclear power plant operation,

vehicle operation and anaesthesiology (Endsley, 1995, Gaba et al., 1995, Hogg et al., 1995, Sollenberger and Stein, 1995, Bolstad, 2000).

There are three main approaches to measuring SA:

- Objective measures: comparing an individual's perception of a situation with the ground truth (during or after a task is performed) (Jones and Endsley, 2000)
- Subjective measures: rating an individual's observed SA either by the individual him/herself or by experienced observers using questionnaires or rating techniques such as PSAQ (Strater et al., 2001) or SART (Taylor, 1989).
- Performance or behavioural measures: SA is inferred from an individual's performance or actions; see (Endsley, 1995) for a discussion of some limitations of this approach.

Another phenomenon studied in the context of human error is mental workload (Wickens and Kessel, 1979, Hancock and Meshkati, 1988, Harris et al., 1995), which is also related to the operator's ability to process information properly, in particular for activities that require high degrees of attention and vigilance. As noted in (Sheridan, 2008), a mental workload measure is deemed a good predictor of human error - better even than a measure of the performance itself of a socio-technical system. Similarly to SA, techniques to assess mental workload include (Young and Stanton, 2004):

- primary task and secondary task performance measures
- physiological measures (e.g. heart rate, eye movement, brain activity, and other measures that may be affected by increases or decreases of mental workload)
- subjective rating techniques (via a variety of scales and questionnaires)

Areas where mental workload has been studied include: aviation and air traffic management (Warm et al., 1998, Wickens, 2002, Loft et al., 2007), car driving (Lemercier and Cellier, 2008) and physiological monitoring in health care (Gorges and Staggers, 2008).

14.4. Studies of Automation Bias and Related Phenomena

Here we outline a line of work which involves the measurement of human performance in the area of "computerised decision support" for human experts. Literature on human factors in engineering and computing has long recognized that even well-designed automation may fail to deliver the improvements it promised, due to initially unexpected effects on people, as reflected in the so called "ironies of automation" (Bainbridge, 1983). We focus here on

unexpected (and undesirable) effects of automation on human performance labelled as "automation bias" by some researchers during the last decade.

As early as 1985, Sorkin & Woods (Sorkin and Woods, 1985) using signal detection theory concluded that in a monitoring task performed by a human-machine system (a person assisted by a computer), optimizing the computer's performance as though it had to perform the task by itself would not always optimize the performance of the human-computer system. A human-machine system can only be optimized or improved as a whole: improving the automation alone or human training alone may be ineffective or wasteful.

“Automation bias” (a term first introduced in the late 1990s) refers to those situations where a human operator makes more errors when being assisted by a computerized device than when performing the same task without computer assistance (Mosier et al., 1998, Skitka et al., 1999). Similar or related concepts are automation-induced “complacency” (Wiener, 1981, Singh et al., 1993, Azar, 1998) and “overreliance” on automation (Parasuraman and Riley, 1997).

Evidence for these phenomena has come from a variety of sources, including anecdotal evidence (Parasuraman and Riley, 1997), observation of professionals in simulated environments (e.g. pilots in a simulated flight) (Mosier et al., 1998), or studies of laymen or professionals in controlled experimental settings (Skitka et al., 1999, Alberdi et al., 2004).

The term “complacency” is often used to account for many of the automation bias phenomena. But such accounts imply value judgments on the human experts. As pointed out by (Moray, 2003), the claim that automation fosters complacency implies that operators are at fault, when, in fact, one also needs to consider the characteristics of the automated tools or the design of the human-machine interaction. Similar conclusions have been obtained in e.g. studies of automation bias in healthcare applications, such as the use of computer aided detection (CAD) for breast cancer screening (Alberdi et al., 2005).

14.5. The Resilience Engineering Movement and Human Performance

Research in system resilience is increasingly noting the positive contributions of humans to system dependability, reflecting the conventional wisdom developed, for example, in the armed forces, reflected in their mistrust in automation in crucial roles (Fitzgerald, 1999). This positive contribution from humans is especially acknowledged in collaborative environments, where people work as part of teams or social networks, whose flexible and adaptable

behaviour can compensate for some of the limitations of the computer tools (Hollnagel et al., 2006). This approach also tends to question the traditional HRA view of the humans as being merely "components" of a system, and instead presents a view of humans as complex systems exhibiting emerging behaviour not susceptible to conventional analyses which, as noted earlier, some researchers find reductive.

Resilience engineering (with its shift of focus from operator error to wider organisational factors) is still in its early stages of development and analyses using this approach are still largely qualitative, rather than quantitative (Sheridan, 2008).

There are, nevertheless, quantitative studies that look at recoverability aspects and at complex human roles in a socio-technical system.

In this section, we can include lines of work, which although not necessarily being part of the resilience engineering movement are nevertheless in tune with some of its principles. For example, work by Brown and colleagues (Brown et al., 2002, Brown et al., 2004) tackles the difficult task of building a human-aware dependability benchmark. This work has addressed, as a case study, the quantitative evaluation of human-assisted recovery tools for internet/e-mail services, including in the assessment the contribution of human operators to the recovery process.

In a related line of work, Vieira and Madeira (Vieira and Madeira, 2002) present an experimental approach for benchmarking the performance and recovery of a Database Management System (DBMS), which includes analysis of the human performance, in terms of the faults of the administrators of the DBMS and their contributions to recoverability.

14.6. Simulation Studies

We present here pointers to relevant pieces of work, which involve the measurement of human reliability in simulated settings.

We have already mentioned work by Maxion and Reeder (Maxion and Reeder, 2005) that collected quantitative data about human performance through ad hoc laboratory experimentation in the evaluation of user-interfaces.

We should also mention the measurement of expert performance in simulated environments, for example, in air traffic control (Sollenberger and Stein, 1995, Brooker, 2006, Leva et al., 2009), and health care (Alberdi et al., 2001, Alberdi et al., 2005).

We can also highlight work at NASA on simulation systems such as MIDAS (Tyler et al., 1998), which uses models of human performance to evaluate candidate crew procedures, controls, displays etc., at early stages of the design of human-machine systems in aeronautics.

An importance piece of work worth mentioning here is recent work by the Halden Reactor Project, in particular, The International HRA Empirical study, whose goal is to benchmark a diverse set of HRA methods against simulator data; preliminary outcomes of this international study have been recently reported at PSAM9, e.g. (Dang et al., 2008a, Dang et al., 2008b, Forester et al., 2008, Parry et al., 2008).

15. Resilience, Fault Tolerance, Resilience Engineering and its Assessment

The word "resilience" has become popular in recent years in the area of ICT and ICT policy. Without reviewing in detail its multiple uses, it is useful to recognise how technical problems and debates in different areas of application are related, highlighting similarities and differences in the problems they pose for quantitative assessment, measurement and benchmarking.

The word "resilience", from the Latin verb *resilire* (re-salire: to jump back), means literally the tendency or ability to spring back, and thus the ability of a body to recover its normal size and shape after being pushed or pulled out of shape, and therefore figuratively any ability to recover to normality after a disturbance. Thus the word is used technically with reference to materials recovering elastically after being compressed, and also in a variety of disciplines to designate properties related to being able to withstand shocks and deviations from the intended state and go back to a pre-existing, or a desirable or acceptable, state. Other engineering concepts that are related to resilience therefore include for instance fault tolerance, redundancy, stability, feedback control.

A review of uses of the word "resilience" by scientists [ReSIST 09] identified uses in child psychology and psychiatry, ecology, business and industrial safety. In many cases, this word is used with its general, everyday meaning. Some users, however, adopt specialised meanings, to use "resilience" as a technical term.

The premise for calling for an everyday word to be used with a new specialised meaning is that there is a concept that needs to have its own name, for convenience of communication, and lacks one. The concept is sometimes a new one ("entropy", for instance), or a new refinement of old concepts ("energy", for instance), or just a concept that needs to be referred to more often than previously (because the problems to be discussed have evolved) and thus requires a specialised word. Sometimes, the motivation is that words previously used for the same concept have been commandeered to denote, in a certain technical community, a more restricted meaning: for instance, after the word "reliability" acquired a technical meaning that was much more restrictive than its everyday meaning, the word "dependability" came to be used, by parts of the ICT technical community, to denote the everyday meaning of "reliability" [Avizienis 04]. For "resilience", a tendency has been to use it, in each specific community, to indicate a more flexible, less prescriptive approach to achieving dependability,

compared to common practices in that community. Thus the above ReSIST document, for instance, concluded that a useful meaning to apply to "resilience" for current and future ICT is "ability to deliver, maintain, improve service when facing threats and evolutionary changes": that is, the important extension to emphasise in comparison with words like "fault tolerance" was the fact that the perturbations that current and future systems have to tolerate include change. While existing practices of dependable design deal reasonably well with achieving and predicting dependability in ICT systems that are relatively closed and unchanging, the tendency to making all kinds of ICT systems more interconnected, open, and able to change without new intervention by designers, is making existing techniques inadequate to deliver the same levels of dependability. For instance, evolution itself of the system and its uses impairs dependability: new components "create" system design faults or vulnerabilities by feature interaction or by triggering pre-existing bugs in existing components; likewise, new patterns of use arise, new interconnections open the system to attack by new potential adversaries, and so on [ReSIST 07]. A document on "infrastructure resilience" [McCarthy 07] identifies "resilience" as an extension of "protection", questioning whether burying cables to prevent hurricane damage is "resilience" but suggesting that installing redundant cabling is.

An important specialised use of the word "resilience" has emerged with "resilience engineering", a movement, or a new sub-discipline, in the area of safety (or, more generally, performance under extreme conditions) of complex socio-technical systems. Here, the word "resilience" is meant to identify enhanced ability to deal with the unexpected, or a more flexible approach to achieving safety than the current mainstream approaches. The meaning is somewhat different between authors, which need not cause confusion if we consider "resilience engineering" as the actual neologism, designating an area of studies and the ongoing debate about it. This area will be further discussed below. From the viewpoint of the problems of quantitative assessment, measurement and benchmarking, the goals of these activities and the difficulties they present, there is no sharp boundary between the socio-technical systems that are of concern to ICT specialists and those addressed by "resilience engineering". There are undoubtedly differences in the typical scales of the systems considered, but the progress in ICT towards the future Internet and greater interconnection of ICT with other infrastructures and activities are cancelling these differences [ReSIST 07]. Most dependability problems in ICT have always involved some social and human factors influencing dependability for instance through design methods and constraints, or through the

maintenance or use of technical systems. In this sense, ICT dependability is about socio-technical systems. As ICT becomes more pervasive and interlaced with human activities, the dependability of the technical components in isolation may become a minor part of the necessary study of dependability and thus of resilience. For example, this occurs in a hospital or air traffic control system, where automated and human tasks interact, and contribute redundancy for each other, on a fine-grain scale. It also occurs where large scale systems involve networks of responsibilities across multiple organisations, as in the provision of services (possibly through open, dynamic collaboration) on the present or future Internet.

Thus, this short survey, written from the vantage point of practices in the technical side of ICT dependability assessment, tries to emphasise the possible new problems, or desirable new viewpoints, that may come from the progressive extension of the domain that ICT specialists have to study towards systems with a more important and more complex social component.

15.1. The "resilience engineering" movement

The title "resilience engineering" has been adopted recently by a movement, or emerging discipline or community, started around a set of safety experts dealing mostly with complex socio-technical systems, like for instance industrial plant, railways, hospitals. A few symposia have taken place focusing on this topic and books have been published. This movement uses the term "resilience engineering" to designate "a new way of thinking about safety" (http://www.resilience-engineering.org/intro.htm). The focus of these researchers is on moving beyond limitations they see in the now-established forms of the pursuit of safety: too much focus on identifying all possible mechanisms leading to accidents and providing pre-planned defences against them; too little attention to the potential of people for responding to deviations from desirable states and behaviours of the system. Thus the resilience engineering authors underscore the needs for reactivity and flexibility, e.g. "The traits of resilience include experience, intuition, improvisation, expecting the unexpected, examining preconceptions, thinking outside the box, and taking advantage of fortuitous events. Each trait is complementary, and each has the character of a double-edged sword." [Nemeth 08]

In using the term "resilience", there is a range between authors focusing on the resilient *behaviour* of the socio-technical system – its visibly rebounding from deviations and returning to (or continuing in) a desirable way of functioning – and those who focus on the characteristics they believe the system must have in order to exhibit such behaviour, like for instance the cultural characteristics and attitudes in the above quote. This degree of ambiguity

need not cause confusion if we simply use the "resilience engineering" phrase to designate a set of related concerns, rather than "resilience" as a specific technical term. It points, however, at the variety of attributes that are inevitably of interest to measure or predict.

Importantly, authors in "resilience engineering" underscore the difference between "resilience" and "safety", the former being just one of the possible means to achieve the latter. Their concern is often one of balance, as they see excessive emphasis on (and perhaps complacency about the effectiveness of) static means for achieving safety, designed in response to accidents, while they see a need for a culture of self-awareness, learning how things really work in the organisation (real processes may be very different from the designed, "official" procedures), taking advantage of the workers' resourcefulness and experience in dealing with anomalies, paying attention to the potential for unforeseen risks, fostering fresh views and criticism of an organisation's own model of risk, and so on. On the other hand, safety can be achieved in organisations that do not depend on "resilience" in this sense of the word, but on rigid, pre-designed and hierarchical approaches [Hale and Heijer 06].

15.2. The appeal of resilience and fault tolerance

Before discussing issues of measurement and quantitative assessment, it is useful to identify some concepts and historical changes that are common to the various technical fields we consider.

When something is required to operate dependably (in a general sense, which here is meant to include "secure against intentional harm"), the means available for ensuring this dependability include mixes of what in the ICT world are often called "fault avoidance" and "fault tolerance" [Avizienis 04]. The former means making components (including, by a stretch of the word "component", the design of the system, with its potential defects that may cause failures of the system) less likely to contain or develop faults, the latter means making the system able to tolerate the effects of these faults.

Historically, the balance between the two approaches is subject to shifts, as is the level of system aggregation at which fault tolerance is applied. For instance, to protect the services delivered by a computer, a designer may add inside the computer redundant component(s) to form a fault-tolerant computer. Alternatively, the designer of a system using the computer (say, an automated assembly line) might provide a rapid repair service, or stand-by computers to be switched in by manual intervention, or manual controls for operators to take control if the computer fails: all these latter provisions make the control function of the assembly line

fault-tolerant (to different degrees). This is a case of shift from fault tolerance in the architecture of a system component (the computer) to fault tolerance in the architecture of the system (the assembly line).

Fault tolerance (for various purposes, e.g., masking permanently disabled components, preventing especially severe effects of failures[17], recovering from undesired transients) is a normal feature of much engineering design as well as organisation design. Fault tolerance against some computer-caused problems is nowadays a normal feature within computer architecture, but over time, as computers in an organisation or engineered plant become more numerous, the space for forms of fault tolerance "outside the computer" increased. Much of the computer hardware and software is obtained off-the-shelf, meaning that for the organisation achieving great confidence in their dependability may be infeasible or expensive, but on the other hand there is a choice of alternatives for error confinement and degraded or reconfigured operation (relying on mixes of people and computers) if only some of these components fail, and for selectively deploying redundant automation – or people – where appropriate.

Such shifts of balance between fault tolerance and fault avoidance, and across levels of application of fault tolerance, occur over time with changes in technology, system size and requirements. Shifts away from fault tolerance are naturally motivated by components becoming more dependable, or their failure behaviour better known (so that fault tolerance is revealed to be overkill), or the system dependability requirements becoming (or being recognised to be) less stringent. Shifts towards more fault tolerance are often due to the observation that fault avoidance does not seem to deliver sufficient dependability, or has reached a point of diminishing returns, and in particular that good fault tolerance will tolerate a variety of different anomalous situation and faults, including unexpected ones. Thus, fault tolerance for instance often proves to be an effective defence against faults that the designers of components do not know to be possible and thus would not have attempted to avoid.

Examples of these factors recur in the history of computing, and can be traced to some extent through the arguments presented at the time to argue that the state of technology and application demanded a shift of emphasis: for instance in the papers by Avizienis in the 1970s [Avizienis 75] proposing more fault tolerance in computers; those of the "Recovery oriented

[17] Including "system design failures": all components function as specified, but it turns out that in the specific circumstances the combination of these specified behaviours ends in system failure: the system's *design* was "faulty".

Computing" project in the early years of the 21st century (http://roc.cs.berkeley.edu/roc_overview.html) for attention to more dynamic fault tolerance in a system comprising multiple computers and operators. In the area of security, similar reasons motivated arguments for more of a "fault tolerance" oriented approach [Dobson and Randell 86], later reinforced by concerns about the inevitable use of off-the-shelf computers and operating systems [Avizienis 04]. Similar considerations have applied to the proposals for fault tolerance against software faults [Chen and Avizienis 77, Popov et al 00]. Very recently, the call for papers for the "Workshop on Resiliency in High Performance Computing (RESILIENCE 2008)" http://xcr.cenit.latech.edu/resilience2008/ points at how the scaling up of massively parallel computations implies that the likelihood of at least one component failing during the computation has become too high if the computation is not able to tolerate such failures; similar considerations have arisen for the number of components in chips, or networks, etc, repeatedly over the years. For an example in larger systems, we may consider titles like "Moving from Infrastructure Protection to Infrastructure Resilience" [GMU 07], advocating a shift from a perceived over-emphasis on blocking threats before they affect critical infrastructure (e.g., electrical distribution grids) to making the latter better able to react to disruption. All these arguments must rely implicitly on some quantification of the risk involved by each alternative defensive solution, although this quantification is not very visible in the literature.

A related, recurrent line of debate is that advocating more flexible and powerful fault tolerance, in which fault tolerance mechanisms, rather than following narrowly pre-defined strategies, can react autonomously and even evolve in response to new situations, like the human mind or perhaps the human immune system [Abbott 90, Avizienis 00]. Some of the recent "autonomic computing" literature echoes these themes [Huebscher 08]. The trade-off here is that one may have to accept a risk that the fault-tolerant mechanisms themselves will exhibit unforeseen and sometime harmful behaviour, in return for an expectation of better ability to deal with variable, imperfectly known and evolving threats. The challenge is to assess this balance of risks, and to what extent a sound quantitative approach is feasible.

In the social sciences' approach to these problems, observations about the importance of redundancy and flexibility underpin the literature about "high reliability organisations" [Rochlin et al 87] and to some extent about "safety cultures". In this picture, the "resilience engineering" movement could be seen as just another shift in which dynamic reaction (fault tolerance) to anomalies is seen as preferable to prior provisions against them, as a precaution

against unexpected anomalies. Its claim to novelty with respect to the community where it originated is in part a focus on the importance of the unexpected. This summary of course does not do justice to the wealth of specific competence about safety in organisations in the "resilience engineering" literature, or about computer failure, human error, distribution network etc to be found in the other specialised literature mentioned above. Our goal here is to identify broad similarities and differences and their implications on assessment, measuring and benchmarking.

Much current emphasis in "resilience engineering" is about flexibility of people and organisations, not just in reacting to individual incidents and anomalous situations, but also in learning from them and thus developing an ability to react to the set of problems concretely occurring in operation, even if not anticipated by designers of the machinery or of the organisation. There is for instance an emphasis, marking recent evolution in the "human factors" literature, on the importance of understanding work practices as they are, as opposed as to how they have been designed to be via procedures and automations of tasks. The real practices include for instance "workarounds" for problems of the official procedures, and may contribute to resilience and/or damage it by creating gaps in the defences planned by designers and managers. It is appropriate to consider differences identified by "resilience engineering" authors between the "resilience engineering" and the older "high reliability organisation" movement. Perhaps the most cited paper [Rochlin et al 87] from the latter discussed how flight operations on U.S. Navy aircraft carrier achieved high success rates with remarkably good safety. This paper focused on four factors: "Self-Design and Self-Replication" (processes are created by the people involved, in a continuous and flexible learning process), the "Paradox of High Turnover" (turnover of staff requires continuous training and conservatism in procedures – both seen as generally positive influences – but also supports diffusion of useful innovation), "Authority Overlays" (distributed authority allowing local decisions by low-ranking people as well as producing higher level decisions through co-operation and negotiation), "Redundancy" (in the machinery and supplies but also in overlapping responsibilities for monitoring and in built-in extra staffing with adaptability of people to take on different jobs as required). A paper about how "resilience engineering" [Nemeth and Cook] differs from this approach refers to healthcare organisations and states that their culture and lack of budgetary margins severely limit the applicability of the four factors seen as so important on aircraft carriers; it points at the potential for improving

resilience by, for instance, IT systems that improve communication within the organisation and thus distributed situational awareness and ability to react to disturbances.

15.3. Resilience and fault tolerance against the unexpected

We see that a frequently used argument for both fault tolerance (or "resilience", seen as going beyond standard practices of fault tolerance in a given community) in technical systems and more general "resilience" in socio-technical systems is based on these being broad-spectrum defences. Given uncertainty about what faults a system may contain or what external shocks and attacks it has to deal with, it seems better to invest in flexible, broad-spectrum defensive mechanisms to react to undesired situations during operation, rather than in pre-operation measures (stronger components, more design verification) that are necessarily limited by the designers' incomplete view of possible future scenarios; likewise, defensive measures.

This argument can, however, be misleading. It is true that general-purpose redundancy and/or increased resources (or attention) dedicated to coping with disturbances as they arise, or to predicting them, can often deal with threats that designers had not included in their scenarios. But there will also be threats that bypass these more flexible defences, or that are created by them. An example can be found in the evolution of modular redundancy at the level of whole computers. The "software implemented fault tolerance" (SIFT) concept in the 1970s [Goldberg 80], arguably the precursor of much current fault tolerance, responded to the fact that one could affordably replicate entire computations running on separate computers, so that the resulting system would tolerate any failure of any hardware or software component within a single computer (or communication channel). This was certainly a more general approach than either more expenditure on fault avoidance without redundancy, or ad-hoc fault tolerance for foreseen failures of each component in a single computer. It was a more powerful approach in that it may well tolerate the effects of more faults, e.g. some design faults in the assembly of the computer or in its software (thanks to loose synchronisation between the redundant computers [Gray 86]). But the SIFT approach also ran into the surprise of "inconsistent failures": the same loose, redundant organisation that gives the system some of its added resilience makes it vulnerable to a specific failure mode. A faulty unit, by transmitting inconsistent messages to other units, could prevent the healthy majority of the system from enforcing correct system behaviour. To tolerate a single computer failure might require four-fold redundancy (and a design that took into account this newly discovered problem) rather than three-fold as previously believed. This was an unexpected possibility, although now, with experience grown from its discovery, it is easy to demonstrate using a

simple model of how such a system could operate. Other events that may surprise designers may be unexpected hardware failure modes; operators performing specific sequences of actions that trigger subtle design faults; new modes of attack on computer security that "create" new categories of vulnerabilities; threats that bypass the elaborate defences created by design (ultra-high availability systems go down because maintenance staff leave them running on backup batteries until they run out, testing at a nuclear power plant involves overriding safety systems until it is too late, attackers circumvent technical security mechanisms in ICT via social engineering); in short, anything that comes from outside the necessary limiting model of the world that the designer uses. Some such surprises arise from incomplete (perhaps inevitably incomplete) analysis of the possible behaviours of a complex system and its environment (cf the Ariane V first-flight accident [Lions 96], and the now common claim that accidents – at least in "mature" organisations – originate from subtle combinations of circumstances rather than direct propagation from a single component failure). Designers also choose "surprises" to which their systems will be vulnerable, by explicitly designing fault tolerance that will not cope with events considered unlikely.

In the ICT area, it is tempting to see "surprises" as manifestations of designer incompetence, and indeed, in a rapidly evolving field with rapidly increasing markets, many will be ignorant about what for others is basic competence. But there is also a component of inevitable surprises. In other areas of engineering it has been observed that the limits of accepted models and practices are found via failure [Petroski 92, Vincenti 93], usually of modest importance (prototype or component tests showing deviations from model predictions, unexpected maintenance requirements in operation, etc), but sometimes spectacular and catastrophic (the popular textbook examples - the Tacoma Narrows bridge, the De Havilland Comet).

Thus, the argument that a more "resilient" design – more open-ended forms of redundancy – offers extra protection is correct, but when it comes to measurement and assessment there is a difference between threats. There is a range of degrees to which quantification is useful, perhaps best illustrated via examples. For a well known and frequent hardware failure mode, we may be able to trust predictions of its frequency, and thus predict the system reliability gain afforded by a specific redundant design, if some other modelling assumptions are correct. For other forms of failure, we may have very imprecise ideas about their frequency – for instance, this usually applies, at the current state of practice, to software failures in highly reliable systems – and yet, we can decide which designs will tolerate specific failure patterns, and via probabilistic modelling even decide whether a design is more resilient than another

one given certain assumptions. Last, there are surprises that violate our modelling assumptions. Designers can try to reduce them by keeping an open mind, and making the system itself "keep an open mind", but have no indication of how successful they are going to be. In the case of organisations, it may well be, for instance, that organisational choices that improve resilience against certain disturbances will be ineffective or counterproductive against others [Westrum 06].

Insofar as resilience is obtained by making available extra resources, limits on resources demand that designers choose against which threats they will deploy more redundant resources. Limits on resources also recommend more flexible designs, in which these resources can deal with more different challenges. Again, these qualitative considerations demand, to be applicable to concrete decisions, at least rough quantification of the risk and costs of different solutions.

This set of considerations has highlighted many areas where measurement and assessment of resilience or fault tolerance are desirable, and started to evoke a picture of measures that may be useful and the difficulties they may involve. The discussion that follows looks at choices of attributes to measure, and difficulties of measurement and prediction, in some more detail, taking a viewpoint inspired by "hard" quantification approaches in engineering and considering some of the issues created by extension towards more complex socio-technical systems.

15.4. Attributes and possible measures of resilience

In quantitative assessment there are always two kinds of potential difficulties: defining measures that usefully characterise the phenomena of interest; and assessing the values (past or future) of these measures.

About the first difficulty, dependability and resilience are broad concepts encompassing multiple attributes, so that there are multiple possible measures. Below is a summary characterisation of categories of measures related to fault tolerance and resilience, with some discussion of their uses and difficulties in measurement and prediction.

The categories are introduced in terms of "systems" (meaning anything from a small gadget to a complex organisation) that have to behave properly despite "disturbances" (a generic term for component faults inside the system, shocks from outside, overloads, anomalous states, no matter how reached). Then, the discussion touches upon differences between categories of

systems and types of "resilience", as well as common problems that may recommend importing insights from some areas of study to others.

15.4.1. Dependable service despite disturbances

The first category of measures that give information about resilience are simply measures of dependability of the service delivered by a system that is subject to disturbances. The better it worked despite them, the more resilient it was. Indeed, a question is why we would want to measure "resilience" or "fault tolerance" attributes, rather than "dependability" attributes. The former are just means for achieving the latter.

For instance, an availability measure for a function of a system obtained over a long enough period of use in a certain environment (pattern of usage, physical stresses, misuse, attacks etc), will be a realistic assessment of how well that function tolerates, or "is resilient" to, that set of stresses and shocks[18].

This kind of measure is certainly useful when applied to documenting past dependability. It will certainly be useful, for instance, in invoking a penalty clause in a contract, if the availability falls short of the level promised. It will also have some uses in prediction. Suppose that the system is a computer workstation used for well-defined tasks in a relatively unchanging environment. A robust measure of past availability ("robust" may imply for instance repeating the measure over multiple workstations of the same type, to avoid bias from variation between individual instances) will be trusted to be a reasonable prediction of future availability (if the environment does not change). Measures on two types of workstations will be trusted to indicate whether one will offer substantially better availability than the other.

[18] A conceptual problem arises here. To use an example, suppose that two computers are made to operate in an environment with high electronic noise. Of the two, computer A is heavily shielded and mostly immune to the noise. The other one, computer B, is not, and suffers frequent transient failures, but always recovers from them so that correct service is maintained. The two thus prove equally dependable under this amount of stress, but many would say that only B is so thanks to its "resilience". Should we prefer B over A? Suppose that over repeated tests, B sometimes fails unrecoverably, but A does not. Clearly, A's lack of "resilience" is not a handicap. Why then should we focus on assessing "resilience", rather than dependability? Or at least, should we not define the quality of interest (whether we call it "resilience" or not) in terms of "correct behaviour despite pressure to behave incorrectly". An answer might be that the resilience mechanisms that B has demonstrated to have will probably help it in situations in which A's single-minded defence (heavy shielding) will not help. But then the choice between A and B becomes an issue of analysing how much better than A B would fare in various situations, and how likely each situation is. Measures of "resilience" in terms of recovery after faltering are just useful information towards estimating measures of such "dependability in various situations".

15.4.1.1 The difficulty of extrapolation

If we wish to compare systems (workstations, in this example), that have not been operated in the same environment, we will sometimes define a reference load (of usage as well as stresses etc) – a "benchmark" workload and stress load, in the current IT parlance. Here, the broader "resilience" literature has to confront issues that are also evident for strict computer dependability evaluation [Madeira and Koopman 01], but with differences of degree. These can be generally characterised as *limits to the extrapolation of measures* to environments that are different from those where the measures were obtained. If a system copes well in the presence of one type of disturbances but less well with another type, changing the relative weights of these two types of disturbances will change the dependability value to be observed. There will not even be a single indicator of "stressfulness" of an environment, so that we can say if a system exhibited - say - 99% availability under the benchmark stress, it will exhibit ≥99% availability in any 'less stressful" environment. Likewise, we won't be able to trust that if system A is better than system B in the benchmark environment, it will still be more dependable in another environment. An extreme, but not unusual case of the extrapolation problem is the difficulty of predictions about systems that are "one of a kind" (from a specific configuration of a computer system, to a specific ship manned by a specific crew, to a specific spontaneous, temporary alliance of computers collaborating on a specific task in the future internet) or will be exposed to "one of a kind" situations: that is (to give a pragmatic definition), systems or situations for which we have no confidence that the measures taken elsewhere, or at a previous time, will still prove accurate. Again, extreme examples are easily found for the human component of systems: an organisation that appears unchanged in term of staff roles, machinery, procedures, may have changed heavily due to staff turnover, or ageing, or even just the experience accumulated in the meantime (for instance, a period without accidents might reduce alertness). Here arises the first reason for going beyond whole-system dependability measures: they do not produce an understanding of *why* a system exhibits a certain level of dependability in a given environment – how each part of the system succumbed or survived the disturbances, which behaviours of which parts accomplished recovery, why they were effective – which could turn into a model for predicting dependability as a function of the demands and stresses in other environments.

Another problem with extrapolation is often created intentionally, as a necessary compromise. If we want a benchmark to exercise the whole set of defences a system has, we need the environment to "attack" these defences. This may require the benchmark load to condense in

a short time many more stress events than are to be expected in real use; but some aspects of resilience are affected by the frequency of stresses. If the system being "benchmarked" includes people, their alertness and fatigue levels are affected. If it involves slow recovery processes (say, background processes that check and correct large bodies of data), an unrealistically high frequency of disturbances may defeat these mechanisms, although they would work without problems in most realistic environments.

Last, there is the problem of resilience against *endogenous stresses*. These exist in all kinds of systems: a computer may enter an erroneous state due to a software design fault being activated or an operator entering inappropriate commands; a factory may suffer from a worker fainting, or from a fire in a certain piece of machinery; and so on. If we wish a common benchmark to measure resilience against these kinds of disturbance, it will need to include some simulation of such events. But this may produce unfair, misleading measures. Perhaps a computer that has very little tolerance to errors caused by internal design faults has been designed this way for the right reasons, since it has no design faults of the types that it cannot tolerate; the less a computer interface tends to *cause* operator errors, the less the computer needs to tolerate them; the less a factory tends to cause workers to become ill on the job, the less it needs to operate smoothly through such events; etc.

This unfairness has a flip side, though: it allows a benchmark to give at least some information about resilience against the unexpected or unplanned-for disturbances. The benchmark deals with hypothetical situations. What if in a factory where nobody ever becomes ill, one day somebody does? What if the computer does have unsuspected design flaws? Likewise, modern regulations require many safety measures for all systems of a certain kind, irrespective the probability, for a specific system, of the situations in which they would be useful. In these circumstances, a dependability or safety "benchmark" (from a fault injection experiment in a computer to an emergency drill in a factory) verifies that certain precautions are in place, and thus certain stresses are likely be tolerated if they were ever to happen.

15.4.2. Measures of tolerable disturbances

A set of attributes that often allow simple and intuitive measures, and thus is heavily used, is the extent of deviation (or damage or disturbance) that a system can tolerate while still later returning to the desired behaviour or state (or preserve some invariant of its behaviour, e.g. some safety property: choosing different invariants will define different measures).

Thus, in ICT one can state that a certain fault-tolerant computer design can mask[19] (without repair) up to *k* faulty components; or a communication code will detect (or be able to reconstruct the original message despite) up to *t* single-bit errors; or that a user interface will tolerate up to *m* erroneous inputs in one transaction; etc. Likewise, in the world of larger systems, we can rate a ship as being able to self-right from a tilt of so many degrees from the upright position; or a factory's staffing level as being calculated to allow for so many absences without loss of productivity. To generalise, this set of attributes, and their measures, are about how far the object of interest can be pushed without losing its ability to rebound or recover; or how quickly it will rebound, or how closely its state after rebounding will resemble the state before the disturbance. To reason properly about these attributes of a system, it is important to recognise them as separate: system A may be "more resilient" than system B from one of these viewpoints, and "less resilient" from another one; for instance, may be slower in recovering from a disturbance of a certain size, but able to recover from a more extreme disturbance.

A great advantage of this type of measures is that for many ICT systems they are easy to obtain directly from their designs: so long as the implementation matches the design in some essential characteristics, we know that certain fault or disturbance patterns are tolerated. They are also typically robust to the extrapolation problem.

If "measuring" on the design is unsatisfactory (for instance we expect the implementation to have flaws; or the required measure is too complex to calculate), we would rely on observations of the system in operation. There may be difficulties in obtaining enough observations of "disturbances" close to the limit, in knowing where the limit is (for systems that should not be tested to destruction), and in deciding whether the system's resilient reaction is deterministic, that is, whether observing successful recovery from a certain extent of disturbances allows us to infer 100% probability of recovery. Again, socio-technical systems offer the most striking examples of the doubts that can affect estimates of these measures.

A limitation of these "maximum tolerable disturbance" measures, even for systems where they are easy to obtain, is that we may well be interested in characterising how well a system rebounds from *smaller* disturbances. For instance, given a form of fault tolerance that allows for some degradation of service, we may then want to measure not just how far the system can be pushed before failing altogether, but the relationship between the size of disturbances and

[19] "Masking" usually meaning that the externally observed behaviour of the system shows no effect of the fault.

the degradation of performance. For instance, for a network (of any kind) one might measure the residual throughput (or other measure of performance) as a function of the amount of network components lost (or other measure of faults or disturbances); this kind of function has been proposed [Garbin and Shortle 07] for resilience of critical infrastructures, leaving open the question of which single-number characterisation (if any) of these curves would be useful in practice. We will return later to characterisations of resilience as a function rather than a synthetic measure.

15.4.3. Measures of "coverage factors"

Since for most systems of interest the resilient behaviour is non-deterministic in practice[20], we are no longer interested in *whether* the system will rebound from a disturbance but in the *probability* of it successfully rebounding; or perhaps the distribution of the time needed for it to return to a desired state; or other probabilistic measures. Thus in fault-tolerant computing we talk about the "coverage" factor of a fault-tolerant mechanism, and we can talk about the distribution of the latency (time to detection) of a component fault or data error.

Importantly, the probability of recovery will be a function of the type of fault or disturbance that occurred. So, all "coverage" measures have to be defined with respect to some stated type, or mix, of faults or disturbances; and the difficulties of extrapolation that characterised measures of dependability under stress affect, in principle, measures of coverage as well. In particular, the desirability and limits of "benchmark" scenarios apply with coverage factors as well as with measures of dependability.

Subject to these limitations, an advantage of measuring coverage factors is that the measure can often be refined so as to fit into predictive models. If coverage factors are obtained for various kinds of disturbances, or for the individual mechanisms present in a system, predictions of the probability of successfully resilient behaviour and hence of dependability measures for a certain future environment can often be obtained from acceptably simple probabilistic models (from simple weighted sums to dynamic models like Markov chains or Stochastic Activity Networks (depending on which assumptions can be trusted about the system and the disturbances).

15.4.4. Measures of socio-technical resilience

Since we are comparing the understanding of resilience with respect to different categories of systems, and the categorisation above is derived from examples at the simple end of the

[20] That is, including any deterministic system that depends on enough variables that the knowledge we can build about it is only statistic or probabilistic

spectrum, it is useful to compare with proposed measures in the areas of complex socio-technical systems. As an example, in a list of proposed attributes of resilience in socio-technical systems Woods [Woods 06] some more easily amenable to precisely defined measures than others. It is interesting to analyse them with reference to the categories given above. They are:

- "buffering capacity", which is essentially the "extent of tolerable disturbances" as discussed above. The issue may be ho easily this can be captured in practically usable measures;
- "flexibility versus stiffness: the system's ability to restructure itself in response to external changes or pressures". It is not clear how this could be measured. For instance, to measure flexibility in the observed operation of a system, we would need to decide which forms of "restructuring" were actually useful, without the benefit of checking how the crisis would develop if the restructuring had not taken place. So, the literature tends to describe this form of "flexibility" through scenarios or anecdotes;
- "margin: how closely or how precarious the system is currently operating relative to one or another kind of performance boundary"; this has often useful definitions in technical systems, for instance we can define an acceptable maximum load on a network before it goes into congestion, or the minimum required set of functioning components necessary for basic services, while in socio-technical system it is often difficult to identify what terms like "stretched to breaking point" may mean.
- "tolerance: how a system behaves near a boundary – whether the system gracefully degrades as stress/pressure increase or collapses quickly when pressure exceeds adaptive capacity". This has parallels in many technical areas, and certainly in ICT, where "graceful degradation" is a frequent requirement, but for which no textbook, standardised measure exists.

15.4.5. Measuring the supposed determinant factors of resilience

An approach to trying to assess dependability (and resilience) in the face of threats that cannot be predicted in detail relies on identifying factors that are believed to enhance resilience. When dealing with well-understood risks, this exercise may take the form of simple design analysis. In many cases, assessment can rely on the combination of analysing which defensive mechanisms are in place, estimates of their coverage factors, and estimates of the distributions of disturbances to which they will need to react. There are of course difficulties with all these

estimates. But when dealing with the human and social determinants of system behaviour, the conjectured determinant factors of resilience often have a "softer" or at least more complex character. A concern in the "resilience engineering" literature is that "measures of outcomes" may lack predictive power: success in the past is no guarantee of success in the future (due to the extreme extrapolation problems mentioned above). Thus a search for "leading indicators" that can be used to assess future resilience. Lists cited in the literature include for instance:

"Leading indicators sets should be based on: Management commitment, Just culture, Learning culture, Opacity, Awareness, Preparedness and Flexibility. Examples of indicators related to preparedness is "crisis training beyond minimum requirements" and to management commitment is "percentage of overtime". Graboski et al. (2007) identify leading indicators at sharp end (Empowerment, Individual responsibility, Anonymous reporting, Individual feedback, Problem identification, Vessels responsibility) and at organizational level (Organizational structure, Prioritizing for safety, Effective communication). Another approach based on organizational resilience focuses on Commitment, Competence, and Cognizance ("three C's" in Reason and Hobbs, 2003). These three C's' are combined with "four P's": Principles, Policies, Procedures, and Practice." [Herrera and Hoyden 08]

Here we are dealing with attributes that are probably important and have complex effects on how well an organisation will perform under stress, and for which an organisation would need to identify reasonable values and trade-offs. Informed judgements about how '"resiliently" organisations will react to stresses will benefit from considering these "indicators". On the other hand, measures of such attributes seem difficult to invent and, more importantly, predictive models based on such measures are probably infeasible.

Indeed, many authors in the "resilience engineering" literature are wary of attempts at quantification, as liable to oversimplify the issues and divert management effort towards achieving required values of measures that have the "advantage" of concrete measurement procedures but no guaranteed relationship to outcomes. Others have used quantitative modelling for illustration and general insight [Duffey 08], borrowing physics-inspired formalisms for modelling complex systems at a macroscopic level.

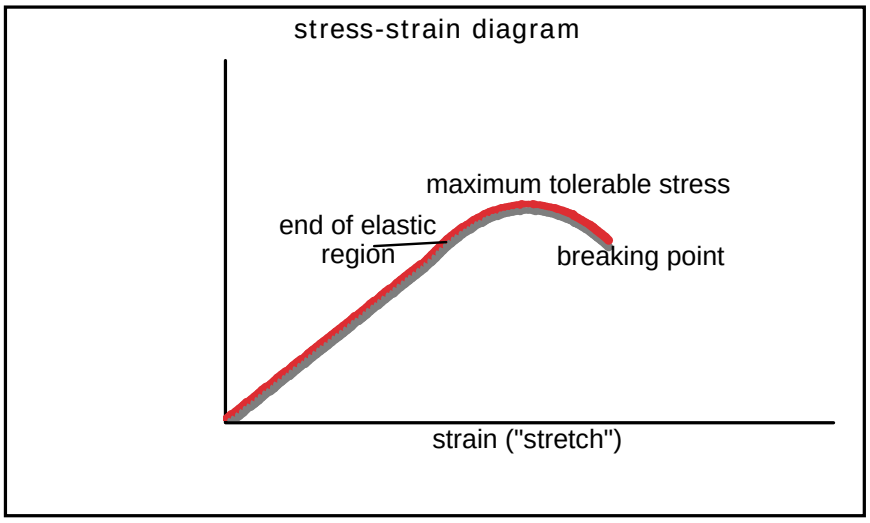
15.4.6. More complex characterisations of resilience - stress-strain curve

Two important topics that have emerged in the discussion so far are: the difference between tolerance/resilience for "design base", expected disturbances and for unexpected or

extraordinary (excluded by design assumption) ones; and the possible need to characterise not just the size of the tolerable stresses, but more detail about the resilient behaviour in response to different levels or patterns of stresses.

In this latter area, one can look for measures like performability [Avizienis 04, Meyer 80], or functions like network throughput as a function of loss of components, which are no sharp departure from dependability modelling approaches that are well established in ICT. While these measures are meaningful, authors have been looking, as exemplified in the previous section, for ways to characterise "resilient" behaviour in a more precise fashion, although accepting that the result may be qualitative insight rather than prediction.

To discuss the various parameters that may characterise resilience in an organisation, Woods and Wreathall [Woods and Wreathall 08] use the "stress-strain" diagram used in material science, as in the figure below. With materials, the *y* axis represents the "stress" applied to a sample of the material (e.g., tensile force stretching a bar of metal), and the *x* axis represents the degree of stretch in the material ("strain"). When tested, the typical building material will exhibit a first regions of linear response (the stretch is proportional to the force applied), followed by a less-than-linear regions and finally by quick yielding that leads to breaking. As it moves from the linear to the sub-linear region, the material also moves from elastic behaviour, where the original size will be regained when the stress is removed, to permanent deformation. A qualitative analogy with organisations is made, in terms of "a uniform region where the organization stretches smoothly and uniformly in response to an increase in demands; and an extra region (x-region) where sources of resilience are drawn on to compensate for non-uniform stretching (risks of gaps in the work) in response to increases in demands". Thus in the "extra region" it is assumed that an organisation that successfully self-modifies shifts onto a new curve, that depart from the now-decreasing main curve and give some extra amount of increase in tolerated "stress" for extra "strain", so as to be able to tolerate stresses beyond its "normal" maximum.



So, this author identifies a region of "orderly" adaptation to increasing stress (in some cases one might identify measures of both stress and strain with an approximately linear relationship, e.g., increased inflow of patients to a hospital being covered by increasing work hours within established procedures). Beyond this maximum, the cost-effectiveness of use of resources decreases and a maximum exists, beyond which extra stress can only be tolerated by some kind of reconfiguration of the organisation, e.g. mustering extra resources or freeing them by changes of operation mode.

This view suggests sets of attributes that can be measured to characterise the response of the system, like the size of the "uniform" range, and the extra stress that can be tolerated before the degeneration into failure. The above author identifies as especially important the ability of an organisation to manage smoothly transitions between regions, and its "calibration", defined as it stability to recognise in which region it is operating, so that reconfiguration is invoked when necessary (and presumably not too often - we note that in many real situations, the ability to assess how well calibrated they were for past decision is limited. One cannot always tell whether a decision to restructure to avoid catastrophic failure was really necessary - especially in view of the uncertainty that the decision maker normally faces in predicting the future). He rightly claims that the stress-strain analogy for organisation behaviour is a first step in clarifying some of the attributes that characterise resilient behaviour (hence also a first step towards quantitative modelling) and importantly highlights the difference between "first-order" and "second-order" adaptive behaviour: the "normal stretching" of the organisation's design in the uniform region, vs the more radical restructuring to work beyond the "normal" limit, but notes the limitation of representing "stress" as a unidimensional attribute, and the need for further work. A limitation that seems important is that this kind of graph implicitly

assumes that the stress-strain relationship can be plotted as independent of time. This matches well those measurement processes for the strength of materials in which stress is increased slowly, moving between states of equilibrium at least up to the maximum of the curve. If the timing of the applied stimulus (as e.g. with sharp impact or repetitive stress) makes a difference in how the material reacts, additional properties can be studied, possibly requiring additional measures. In organisations (or for that matter in computers), many of the stresses may need to be characterised in terms of dynamic characteristics, or need to be defined in practice in terms of timing characteristics of events.

Considering the time factor may also bring into play other aspects of self-stabilisation, and other necessary design trade-offs. For instance, making a ship more "stable" (increasing its metacentric height, so that it will self-right more promptly after heeling to one side) makes it more liable to roll at higher frequency following the tilt of the waves, so that it can reduce the effectiveness of the crew, make a warship unable to use its weapons. Likewise, all "resilience" that relies on detecting (or predicting) component failures or shocks must strike a compromise between the risk of being too "optimistic" – allowing the situation to deteriorate too far before reacting – or too "pessimistic" – reacting too promptly, so that false alarms, or reactions to disturbances that would resolve themselves without harm, become too much of drain on performance or even damage resilience itself.

15.5. Conclusions

A theme running through this survey has been that as fault tolerance (or resilience), that is, dynamic defences, exist in all kinds of systems, the measures that may be appropriate for studying them also belong to similar categories and the difficulties in defining measures, measuring, and predicting the values of measures also belong to common categories. Interest in studying and/or in extending the use of fault tolerance or resilience[21] has expanded of late in many areas, and we can all benefit from looking at problems and solutions from different technical areas. I gave special attention to the "resilience engineering" area of study, since its choice of topic problems highlights extreme versions of measurement and prediction problems about the effectiveness of "resilience" that exist in the ICT area. In all these areas there are spectra of prediction problems from the probably easy to the intractable. The "resilience engineering" movement has raised important issues related to the measurement

[21] U.S. Navy aircraft carriers practiced the use of redundancy long before Rochlin et al studied it. On the other hand their study prompted more organisations to recognise and protect forms of redundancy in their operation and/or to consider applying it.

and prediction of “resilience” attributes. One is simply the recognition of the multi-dimensionality of “resilience”. For instance, Westrum [Westrum 06] writes: “Resilience is a family of related ideas, not a single thing. The various situations that we have sketched offer different levels of challenge, and may well be met by different organizational mechanisms. A resilient organization under Situation I will not necessarily be resilient under Situation III (these situations are defined as having different degrees of predictability). Similarly, because an organization is good at recovery, this does not mean that the organization is good at foresight”.

The boundaries between strict technical ICT systems and socio-technical systems are fuzzy, and for many applications the recognition of social components in determining meaningful assessment of dependability is important [ReSIST 07]. Concerns about improving measurement and quantitative prediction are often driven by the concrete difficulties in applying existing methods in new systems: just as increasing levels of circuit integration and miniaturisation made it infeasible to monitor circuit operation at a very detailed level via simple probes and oscilloscopes, so the deployment of services over large open networks and through dynamic composition may create new difficulties in measuring their dependability. More general problems may arise, however: do we need to choose appropriate new measures for characterising the qualities of real interest? If they are amenable to measurement in practice, to what extent will they support trustworthy predictions? To what extent may the benefit of “reasonably good” measures (perhaps acceptable proxies for the “truly important” ones) be offset by the reaction to their adoption: designers and organisations focusing on the false target of good values of these measures, perhaps to the detriment of the actual goal of dependability and resilience.

These questions underlie all assessment of resilience and dependability, but more markedly so as the socio-technical systems studied become less “technical” and more “social”. Authors in ”resilience engineering” have identified research problems in better characterising, even at a qualitative, descriptive level, the mechanisms that affect resilience. Quantitative measurement may follow. Quantitative predictive models may or may not be feasible, from the abundant research in modelling at various levels of detail the dependability of complex infrastructure and ICT; quantitative approach from mathematical physics may also yield insight even without predictive power [Duffey 08]. Research challenges include both pushing the boundary of the problems that can be addressed by sound quantitative techniques, and finding clearer indicators for these boundaries. There are enough historical examples of quantitative

predictions proving misleading, and perhaps misguided, but we often see these with the benefit of hindsight. Perhaps most important would be to define sound guidance for "graceful degradation" of quantitatively driven decision making when approaching these limits: more explicit guidance for exploiting the advantages of measurement and quantitative prediction "as far as they go" but avoiding potential collapse into unrealistic, "pure theory" driven decisions making.

16. Analysis and Presentation Techniques

It is well known that assessing, measuring, and benchmarking resilience in computer systems is a complex task. In spite of the effort put on the development of adequate tools and the intensive research devoted to the mitigation of key problems such as experiment representativeness, intrusiveness and portability of tools, just to name a few, two important questions remain largely unanswered:

- How to analyze the usually large amount of raw data produced in resilience evaluation experiments, especially when the analysis is complex and have to take into account many aspects of the experimental setup (e.g., target systems, configurations, workload and programs, faultload, diversity of measures, etc)?
- How to compare results from different experiments or results of similar experiments across different systems if the tools, data formats, and the setup details are different and, often, incompatible?

Many evaluation tools such as fault injection and robustness testing tools described in chapters 7 and 13 provide rudimentary means to analyze data or, more frequently, just store the raw results in a spreadsheet format such as Microsoft Excel®. Although this approach can be acceptable for very specific (and simple) analysis, it is clearly not enough when the amount of raw data is very large, the analysis required is complex, and particularly when heterogeneous data from different experiments have to be analyzed and cross-exploited.

This chapter presents two key data analysis and presentation techniques: data warehousing & OLAP, and data mining. These techniques are already being used in the context of dependability assessment and we believe that will be used more and more to analyse large amount of dependability raw data.

16.1. Basic concepts on data warehousing and OLAP

Data warehousing refers to "a collection of decision support technologies aimed at enabling the knowledge worker (executive, manager, or analyst) to make better and faster decisions" [Chauduri 97]. A data warehouse is a global repository that stores large amounts of data that has been extracted and integrated from heterogeneous systems (operational or legacy systems). OLAP (On-Line Analytical Processing) is the technique of performing complex analysis over the information stored in a data warehouse [Chauduri 97]. The data warehouse coupled with OLAP enable decision makers to creatively analyze and understand business

trends since it transforms operational data into strategic decision making information. The following sections present some key concepts on data warehousing and OLAP. The reader interested in knowing more about data warehousing can find a complete course in [Kimball 02].

16.1.1. The multidimensional model

In data warehousing the data is organized according to the multidimensional model, which includes two kinds of data: facts and dimensions. **Facts** are numeric or factual data that represent a specific business or process activity and each **dimension** represents a different perspective for the analysis of the facts. Each dimension is described by a set of attributes. For instance, in the classical example of a chain of stores [Kimball 02] represented in Figure 15.1, some of the dimensions are products, stores, and time, while the facts (the small cubes) contain the total sales, profit, etc for a given product in a given store in a single day. Note that the facts are just numerical quantities and only acquire meaning when are referenced to the dimensions. Obviously, Figure 15.1 is a very simple example. Normally, there are more than three dimensions and the dimension attributes represent a detailed description of the dimensional data (e.g., the description of a product in the store example includes many attributes, such as brand, category, package, etc). The OLAP analysis over a multidimensional cube consists of **dicing** and **slicing** the data in order to compute the desired measures.

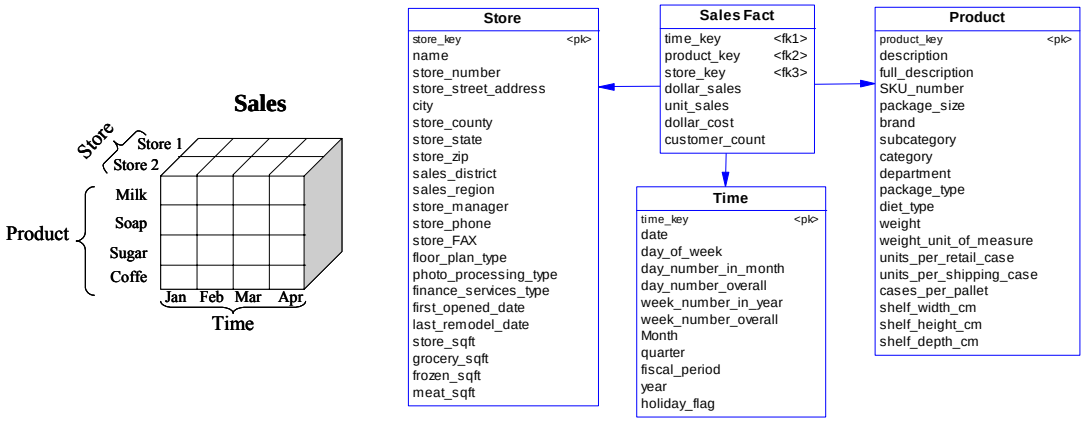


Figure 15.1. Simple example of multidimensional model: logical view (left) and a physical star schema (right).

Although it is possible to store the data warehouse in a multidimensional database server, most of the data warehouses and OLAP applications store the data in a relational database. That is, the multidimensional model is implemented as one or more star schemas [Chauduri

97] formed by a large central fact table surrounded by several dimensional tables related to the fact table by foreign keys (see example in the right side of Figure 15.1).

Typical data warehouses are periodically loaded with new data that represents the activity of the business since the last load (normally the periodical loads are done on a daily basis). This is part of the typical life-cycle of data warehouses. At the end of the day, the data that represents the daily business activity is extracted from the specific systems were it is produced and loaded into the data warehouse. In practice, the raw data came from several sources (all sort of systems meant to support the daily business operations such as database applications, spreadsheets, specific applications, etc) and it is necessary to introduce some transformations to assure data consistency, before loading that data into the data warehouse.

In short, the key elements of a Data Warehousing environment are:

- **Source operational databases** – operational databases and other legacy systems or files from which the operational data is gathered.
- **Loading applications** – Normally the loading applications are specific programs that read the raw data into standard database tables for transformation and final load into the data warehouse.
- **Data warehouse** – The raw data is stored in a multidimensional data model (in the form of a star schema). In order to store data from each specific context it is necessary to do multidimensional analysis in order to identify the required star schema (facts and dimensions).
- **OLAP tool** – The OLAP tools are used to analyze the data and compute the measures.

16.1.2. Major steps

The major steps needed to use business intelligence in practice are the following:

- Multidimensional analysis of the data to be analyzed and **definition of the adequate star schema**. Once the schema is defined, all the tables (fact and dimensions tables) are created in the data warehouse and the star schema is prepared to receive the raw data.
- Data extraction, transformation and loading (known as the ETL process). A general-purpose loading application is used to **define the loading plans** for each table in the star scheme. A loading plan is a kind of script that tells the loading application where to read the data to be loaded into each table of the star schema created in the data warehouse (it is necessary a loading plan per table in the star schema). If the data sources (e.g. the files

storing the data produced during projects) are too specific or have idiosyncrasies that cannot be handled by a general-purpose loading application, then it is necessary to write a specific program to get the data and load it in the star schema table (normally these programs are quite simple as they just transport data from one point to another). Once all the loading plans (or specific loading programs) are prepared, the data warehouse can be loaded with raw data. Every time new data becomes available the corresponding loading plans are run again to add the new data into the data warehouse.

- **Analyze the data with an OLAP tool**. This includes not only the normal computation of measures, but also the detailed analysis to find interesting patterns in the results. This is in fact the main advantage of using OLAP analysis, as finding unexpected results and analyze data trends is the typical job done when OLAP is applied to this classical field (i.e., the business data). For example, the identification of unexpected results and the isolation of the effects of a specific aspect of each dimension (e.g. only results for a specific technology) are the normal routine in OLAP.

It is worth noting that data warehousing and OLAP applications always require some **administration** (in addition to the three main steps mentioned above to implement the data warehouse). Although this requires database administration skills, it is just normal routine to a database administrator. A key aspect is performance tuning, which includes the identification and creation of indexes, aggregates, clusters, etc.

16.2. Data Mining

Data mining [Han 01, Hand 01] is usually defined as an interdisciplinary field bringing together techniques from machine learning, pattern recognition, statistics, databases, and visualization to address the issue of extracting *previously unknown, valid and actionable information* from *large databases* to be used for making business decisions. The need for data intelligence solutions came from the realization that traditional decision-support methodologies, which combine simple statistical techniques with executive information systems, do not scale well to large databases and data warehouses within the time limits imposed by today's business environments. This introduction on data mining (i) outlines the usual architecture of data mining setups, (ii) discusses the key steps of data mining and (iii) presents an overview on data mining techniques.

16.2.1. Architecture of Data Mining

A typical data mining setup consists of three key components (Figure 15.2):

- Data is stored in a conventional database or a *data warehouse*.
- Data intelligence algorithms are implemented in the *data mining engine*. The data mining engine obtains the input data from the database by issuing queries.
- The *user interface* is responsible for issuing mining commands to the data mining engine and translating the abstract results of data mining into easy to understand diagrams, decision trees, etc.

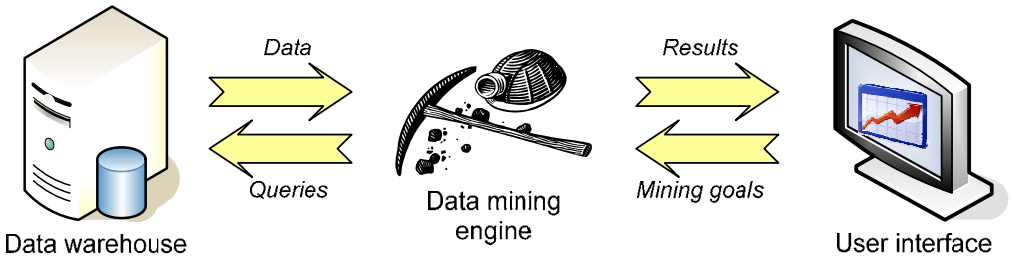


Figure 15.2. Typical architecture of data mining

16.2.2. Process of Data Mining

Data mining is an iterative process consisting of (i) data selection, (ii) data preparation, (iii) data mining and (iv) result analysis steps [IBM 99]:

- The **data selection** step involves the identification and selection of all relevant data sources (database artefacts, log files, etc.) that can be used as input for data mining. E.g., a vehicle retail company may want to understand the long-term behaviour of its customers; in this case, the customer information database and selling records are likely to be considered as inputs for data mining.
- The **data preparation** step involves the treatment of missing values, outliers, and the possible introduction of calculated attributes. Specific data mining algorithms may have their own data preparation requirements; data preparation could also include data reduction, which is defined by the maximum number of variables that an algorithm can effectively utilize. E.g., in case of the vehicle retail company above, if the birth dates of customers and purchase dates of various products (e.g., bicycles, motorbikes, cars, etc.) are stored in the input database, it is probably a good idea to introduce a calculated attribute indicating the age of the customer at the occurrence date of specific purchases; on the other hand if the actual data mining goal focuses on individual customers, it is probably a good idea to remove corporate customers from the data set because some very

large scale transactions (e.g., buying one hundred cars) are likely to be such outliers that may distort the results.

- The actual **data mining** step involves the execution of the various data mining algorithms against the prepared data sets. E.g., the vehicle retail company above may be interested in the long term shopping habits of customers over the years, thus decides to run a sequential pattern discovery algorithm on the data set to understand the chronological relations of bicycle, motorbike and car purchases of customers.
- The **results analysis** step aims at documenting and explaining results of the data analysis with non-technical terms enabling executives to draw business decisions based on the newly extracted information. This documentation should describe all data attributes involved in the resulting model and the human-readable explanation of the model built. E.g., in case of the vehicle retail company above the sequential pattern discovery may indicate that customers between 10 and 15 years mostly buy bicycles, then between 15 and 20 are likely to purchase a motorcycle and above 20 they buy their first car – in this case the newly introduced "age" attribute has to be documented and the suggested timeline of purchases presented.

16.2.3. Data Mining Techniques

Data mining methods are usually assigned to three families: predictive modeling, database segmentation and link analysis [Michie 94, Bigus 96, IBM 99]; another interesting application of data intelligence is text mining [Feldman 06].

Predictive modeling resembles the human learning experience, where we learn how to classify real-world objects into abstract categories by identifying the essential underlying characteristics of phenomena amongst the possibly high number of less important attributes. The goal of predictive modeling is to build similar models by analyzing a *teaching data* set and identifying the attributes and their relations that represent the key factors for classifying database records using statistical methods. Obviously the teaching data set must include complete, valid observations from which the model can learn how to make accurate predictions (this model construction scheme is called *supervised learning*). Models are developed in two phases: training and testing. Training refers to building a new model by using historical data, and testing refers to trying out the model on new, previously unseen data to determine its accuracy and physical performance characteristics. Two specializations of predictive modeling are classification and value prediction. *Classification* aims at assigning

each record in the database to one from a finite set of pre-determined classes. *Value prediction* aims at building a model for estimating a continuous numeric value.

Database segmentation aims at partitioning a database in segments of similar records i.e., ones that share a number of properties and so are considered to be homogeneous. With respect to the *star scheme* of the data warehouse, the database segmentation process means typically the identification of similar records in a *dimension table*. As database segmentation can segment a database without any apriori information about the number or type of segments, its operation is considered to be *unsupervised learning*. Two specializations of database segmentation are demographic clustering and neural clustering. *Demographic clustering* operates on records with attributes that represent the record belonging to some category; these algorithms identify groups of similar records using distance measures based on a voting principle. *Neural clustering* operates on records with continuous numeric attributes; these algorithms use neural networks for building the segmentation model.

Link analysis aims at establishing links (associations) between individual records or sets of records in the database. With respect to a *warehouse organization*, link analysis is performed on a dimension table that has a field that identifies which group the individual records belong to. Three specializations of link analysis are association discovery, sequential pattern discovery and similar time sequence discovery. *Association discovery* aims at identifying record types (typically product types) that are likely to belong to the same transaction (i.e., products sold together). *Sequential pattern discovery* aims at identifying typical patterns over sequences of records that are labelled with a date attribute and are tied together by some common identifier (typically purchases performed by the same customer). *Similar time sequence discovery* aims at discovering links between two time dependant data sets.

Another important application field of data intelligence is **text mining**. Text mining aims at extracting high quality information from text, where quality is usually measured by a combination of relevance, novelty and interestingness. Typical application areas of text mining are text categorization, document summarization and sentiment analysis.

16.3. Application of Data Intelligence on Dependability Field

[Madeira 03] proposes a new approach, based on multidimensional analysis and data warehousing & OLAP (On-Line Analytical Processing) technology, to solve the problem of analyzing, sharing, and cross-exploiting results from dependability evaluation experiments. The central idea is to collect the raw data produced in dependability evaluation experiments

and store it in a multidimensional data structure (data warehouse). The data analysis is done through the use of commercially available OLAP tools such as the ones traditionally used in business decision support analysis [Kimball 02] (e.g., Discoverer® from Oracle). That is, instead of following the usual trend of adding data analysis features to fault injectors and robustness testing tools, [Madeira 03] proposes a clear separation between the experimental setup (target specific) and the result analysis setup (general in our approach). Existing tools and experimental setups are used as they are and just export the data obtained in the experiments to a data warehouse, where all the analysis and cross-exploitation of results can be done in an efficient and general way.

In [Pintér 05] it is proposed a novel approach for identifying the key infrastructural factors determining the behavior of systems in the presence of faults by the application of *intelligent data processing methods* that have already been successfully applied in the business field for extracting previously unknown knowledge from large databases. The key idea of the approach is to perform benchmarking experiments on multiple configurations by applying different implementations of the same COTS component (e.g., different hardware setups, operating systems) and record as much information as possible about the infrastructure and the delivered performance and dependability attributes. On the basis of this information *data mining experiments* are carried out to identify which infrastructural factors were really relevant enabling the developers to improve the system without a-priori assumptions.

[Pintér 08] discusses how data warehousing, OLAP, and data mining can be integrated for the analysis of dependability-related data.

The application of data warehousing and data mining in the automatic processing of data obtained from dependability experiments has received increasing attention recently. In fact, these techniques were shown to provide beneficial support for the automatic identification of key factors determining performance and dependability attributes of complex HW/SW systems. Nevertheless, further research work is still needed to spark the use of these technologies in a systematic manner during resilience assessment, measurement and benchmarking.

References

[Abbott 90] R. J. Abbott, "Resourceful systems for Fault Tolerance, Reliability, and Safety," *ACM Computing Surveys*, vol. 22, pp. 35-68, 1990.

[Acunetix 07] Acunetix Ltd, February 12, 2007, http://www.acunetix.com/news/security-audit-results.htm

[Adelard] See http://www.adelard.com/web/hnav/resources/iee_pn/index.html

[Adve et al. 00] V. S. Adve, R. Bagrodia, J. C. Browne, E. Deelman, A. Dube, E. Houstis, J. Rice, R. Sakellariou, D. Sundaram-Stukel, P. J. Teller, and M. K. Vernon, "Poems: End-to-end performance design of large parallel adaptive computational systems", IEEE Transactions on Software Engineering, Special Section of invited papers from the WOSP '98 Workshop, 26(11):1027-1048, November 2000.

[Aidemark 01] J. Aidemark, J. Vinter, P. Folkesson, J. Karlsson, "GOOFI: Generic Object-Oriented Fault Injection Tool", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2001, Göteborg, Sweden, July 2001, pp. 83-88.

[Alberdi et al 2001] Alberdi, E., Becher, J. C., Gilhooly, K., Hunter, J., Logie, R., Lyon, A., McIntosh, N. & Reiss, J. (2001) Expertise and the interpretation of computerized physiological data: implications for the design of computerized monitoring in neonatal intensive care. *International Journal of Human-Computer Studies,* 55**,** 191-216.

[Alberdi et al 2004] Alberdi, E., Povyakalo, A. A., Strigini, L. & Ayton, P. (2004) Effects of incorrect CAD output on human decision making in mammography. *Acad Radiol,* 11**,** 909-918.

[Alberdi et al 2005] Alberdi, E., Povyakalo, A. A., Strigini, L., Ayton, P., Hartswood, M., Procter, R. & Slack, R. (2005) Use of computer-aided detection (CAD) tools in screening mammography: a multidisciplinary investigation. *Br J Radiol,* 78**,** S31-40.

[Alexander, K. and Ludwig, H. 2003]. "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services." Journal of Network and Systems Management 11(1): 57-81.

[AlTurki 2009] Probabilistic Modeling and Analysis of DoS Protection for the ASV Protocol, Musab AlTurki, Jose Meseguer, and Carl A. Gunter. Electronic Notes in Theoretical Computer Science 234, pages 3-18, 2009.

[Alvarez 97] G. A. Alvarez and F. Cristian, "Centralized failure injection for distributed, fault-tolerant protocol testing," in 17th Int. Conf. on Distributed Computing Systems, 1997, pp. 78-85.

[AMSD1] AMSD. Dependable Embedded Systems Roadmap. AMSD Deliverable D2.1. May 2003.

[AMSD2] AMSD. AMSD: A Dependability Roadmap for the Information Society in Europe Part 1 – An Insight into the Future. AMSD Deliverable 1.1 – Road-map.

[AMSD3] AMSD. AMSD: A Dependability Roadmap for the Information Society in Europe Part 2 – Appraisal of related IST Roadmaps. Deliverable 1.1 – Road-map.

[AMSD4] AMSD. AMSD: A Dependability Roadmap for the Information Society in Europe Part 3 - Towards a Dependability Roadmap. AMSD Deliverable 1.1 – Road-map.

[Anderson 2008] Ross J Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley, 2008.

[Anderson & Moore 2007] R. Anderson and T. Moore, "The Economics of Information Security: A Survey and Open Questions", in *Fourth Bi-Annual Conference on the Economics of the Software Internet Industries,* (Toulouse, France), 2007.

[Ando 08] Ando, H., Kan, R., Tosaka, Y., Takahisa, K., and Hatanaka, K., "Validation of Hardware Error Recovery Mechanisms for the SPARC64 V Microprocessor." 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008), pp. 62-69, 2008.

[Andrés 06] D. d. Andrés, J. C. Ruiz, D. Gil, and P. Gil, "Run-Time Reconfiguration for Emulating Transient Faults in Vlsi Systems," in Int. Conf. on Dependable Systems and Networks 2006, pp. 291-300.

[Andrés 08] D. de Andrés, J. C. Ruiz, D. Gil, and P. Gil, "Fault Emulation for Dependability Evaluation of VLSI Systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 4, April 2008.

[Anon 85] Anon et al, "A Measure of Transaction Processing Power", Datamat, April 1, 1985.

[Antoni 03] L. Antoni, R. Leveugle, and B. Feher, "Using Run-Time Reconfiguration for Fault Injection Applications," IEEE Transactions on Instrumentation and Measurement, vol. 52, pp. 1468-1473, 2003.

[Apple 2006] "Apple Crash Reporter" accessed Jan. 2009, from http://developer.apple.com/technotes/tn2004/tn2123.html

[Arafat, O., A. Bauer, et al. 2005]. Runtime Verification Revisited. München, Technischen Universität München.

[Arkin 05] B. Arkin, S. Stender, and G. McGraw, "Software Penetration Testing" IEEE Security & Privacy, Ed. Gary McGraw, IEEE Computer Society, 2005.

[Arlat 90] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, E. Martins, and D. Powell, "Fault Injection for Dependability Validation — a Methodology and Some Applications," IEEE Transactions on Software Engineering, vol. 16, pp. 166-182, February 1990.

[Arlat 02] J. Arlat, J.-C. Fabre, M. Rodriguez, F. Salles, "Dependability of COTS Microkernel-Based Systems", IEEE Transactions on Computers, Feb. 2002, Vol. 51, No. 2, pp.128-163

[Arlat 03] J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs, and G. H. Leber, "Comparison of Physical and Software-Implemented Fault Injection Techniques," IEEE Transactions on Computers, vol. 52, pp. 1115-1133, 2003.

[Arlat 93] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie and D. Powell, "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems", IEEE Transactions on Computers, Vol. 42 (8), pp 913-923, 1993.

[Assaf 04] M. Assaf, S. Das, E. Petriu, L. Lin, C. Jin, D. Biswas, V. Groza, and M. Sahinoglu, "Hardware and software co-design in space compaction of digital circuits", in IEEE Instrumentation Measurement Tech. Conf., vol. 2, pp. 1472-1477, 2004.

[AutonomicIBM] IBM Autonomic Computing Initiative, http://www.research.ibm.com/autonomic/.

[Avizienis 75] A. Avizienis, "Fault-Tolerance and Fault-Intolerance: Complementary Approaches to Reliable Computing," presented at International Conference on Reliable Software, Los Angeles, California, 1975.

[Avizienis 00] A. Avizienis, "Design Diversity and the Immune System Paradigm: Cornerstones for Information System Survivability," presented at Third Information Survivability Workshop (ISW-2000), Boston, Massachusetts, USA, 2000.

[Avizienis 04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Trans. on Dependable and Secure Computing, 1(1):11–33, 2004.

[Avresky 96] D. Avresky, J. Arlat, J. C. Laprie, and Y. Crouzet, "Fault Injection for Formal Testing of Fault Tolerance," IEEE Transactions on Reliability, vol. 45, pp. 443-455, 1996.

[Ayache, J. M., P. Azema, et al. 1979]. Observer, a Concept for On-Line Detection for Control Errors in Concurrent Systems. Ninth International Symposium on Fault-Tolerant Computing, Madison.

[Azar 1998] Azar, B. (1998) Danger of automation: It makes us complacent. *APA monitor,* 29**,** 3.

[Bailey *et al.* 2005] M. Bailey, E. Cooke, F. Jahanian and J. Nazario, "The Internet Motion Sensor - A Distributed Blackhole Monitoring System", in *Network and Distributed Systems Security Symposium (NDSS-2005),* (San Diego, CA, USA), 2005.

[Bainbridge 1983] Bainbridge, L. (1983) Ironies of Automation. *Automatica,* 19**,** 775-779.

[Baier 99] C. Baier, J.-P. Katoen, and H. Hermanns. "Approximate symbolic model checking of continuous time Markov chains". In Proceedings of CONCUR'99, volume 1664 of LNCS, pages 146—162, 1999.

[Balbo 01] G. Balbo, "Introduction to stochastic petri nets", Lectures on Formal Methods and Performance Analysis, volume 2090 of Lecture Notes in Computer Science, pages 84-155. Springer Verlag, 2001.

[Banbury and Tremblay 2004] Banbury, S. and Tremblay, S. (2004) *A cognitive approach to situation awareness: Theory, measurement and application,* Aldershot, UK, Ashgate Publishing.

[Bar-El 06] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, C.Whelan, "The Sorcerer's Apprentice Guide to Fault Attacks," Proceedings of the IEEE , vol.94, no.2, pp.370-382, Feb. 2006

[Barbosa 07] R. Barbosa, N. Silva, J. Durães, H. Madeira, "Verification and Validation of (Real Time) COTS Products using Fault Injection Techniques", 6th IEEE International Conference on COTS-based Software Systems, Alberta, Canada, February 2007

[Baresi 02] L. Baresi and R. Heckel, "Tutorial introduction to graph transformation: A software engineering perspective", in Proc. Int. Conf. on Graph Transformation (ICGT), LNCS 2505, pp 402-439, Springer, 2002.

[Barger 03] P. Barger, J.-M. Thiriet, M. Robert, "Dependability Analysis of a Distributed Control or Measurement Architecture," Proc. of IEEE IMTC 2003 - Instrumentation and Measurement Technology Conference, Vail, CO, USA, 20-22 May 2003

[Barton 90] J. H. Barton, E. W. Czeck, Z. Z. Segall, and D. P. Siewiorek, "Fault Injection Experiments Using Fiat," IEEE Transactions on Computers, vol. 39, pp. 575-582, 1990.

[Basra and Kirwan 1998] Basra, G. and Kirwan, B. (1998) Collection of offshore human error probability data. *Reliability Engineering & System Safety,* 61**,** 77-93.

[Bause et al. 98] F. Bause, P. Buchholz, and P. Kemper. A toolbox for functional and quantitative analysis of deds. In Lecture Notes in Computer Science, number 1469, pages 356-359. R. Puigjaner, N. N. Savino, and B. Serra, 1998.

[Beautement et al. 2008] A. Beautement, M. A. Sasse and M. Wonham. "The Compliance Budget: Managing Security Behaviour in Organisations". NSPW 2008, Olympic, California, USA. Sep 2008.

[Bello and Colombari 1980] Bello, G. C. and Colombari, V. (1980) The human factors in risk analyses of process plants: The control room operator model 'TESEO'. *Reliability Engineering,* 1**,** 3-14.

[Benso 03] A. Benso and P. Prinetto, Fault injection techniques and tools for embedded systems reliability evaluation. Boston: Kluwer Academic Publishers, 2003.

[Béounes et al. 93] C. Béounes et al. Surf-2: A program for dependability evaluation of complex hardware and software systems. In Proc. of the 23rd Int. Symp. on Fault-Tolerant Computing, pages 668-673, Toulouse, France, 1993.

[Bigus 96] Joseph P. Bigus. Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[BIMP 08] BIMP, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. ISO International Vocabulary of Basic and General Concepts and Associated Terms (VIM), 2008.

[Bishop 90] P G Bishop, Dependability of Critical Computer Systems 3: Techniques Directory, Elsevier Applied Science, ISBN 1-85166-544-7, 1990.

[Bishop and Bloomfield 95] P G Bishop and R E Bloomfield, The SHIP Safety Case. In proceedings, SafeComp 95, 14th IFAC Conf. on Computer Safety, Reliability and Security (ed. G. Rabe), Belgirate, Italy, 11-13 October 1995, Springer, ISBN 3-540-19962-4., Adelard 1995.

[Bishop and Bloomfield 98] P G Bishop and R E Bloomfield, A Methodology for Safety Case Development. Safety-critical Systems Symposium, Birmingham, UK, Feb 1998 in Industrial Perspectives of Safety-critical systems, ISBN 3-540-76189-6, Springer-Verlag 1998.

[Bishop and Bloomfield 02] P G Bishop and R E Bloomfield, "Worst Case Reliability Prediction Based on a Prior Estimate of Residual Defects", Thirteenth International Symposium on Software Reliability Engineering (ISSRE '02), November 12-15, Annapolis, Maryland, USA, pp. 295-303, 2002.

[Bishop et al 02a] P G Bishop, R E Bloomfield, and P K D Froome, "Justifying the use of software of uncertain pedigree (SOUP) in safety related applications", 5th International Symposium: Programmable Electronic Systems in Safety Related Applications, Cologne, 7-8 May, 2002.

[Bishop et al 02b] P G Bishop, R E Bloomfield, T P Clement, and A S L Guerra, "Software Criticality Analysis of COTS/SOUP", SAFECOMP 2002, 10-13 September 2002, Catania, Italy, pp. 198-211, 2002.

[Bitton 83] Dina Bitton , David J. DeWitt , Carolyn Turbyfill, "Benchmarking Database Systems – A Systematic Approach", 9th International Conference on Very Large Data Bases, p.8-19, October 31-November 02, 1983.

[Bloomfield 90] R E Bloomfield, "SafeIT, The Safety of Programmable Electronic Systems: a government consultation document on activities to promote the safety of computer-controlled systems", UK Department of Trade and Industry, May, 1990.

[Bloomfield and Brazendale 90] R E Bloomfield and J Brazendale, "SafeIT2, Standards framework", UK Department of Trade and Industry, June 1990.

[Bloomfield and Guerra 02] R E Bloomfield, and A S L Guerra, "Process Modelling to Support Dependability Arguments", Proc DSN 2002, IEEE Computer Society, Washington DC USA, 2002.

[Bloomfield and Littlewood 03] R E Bloomfield and B Littlewood, "Multi-legged Arguments: The impact of Diveristy upon Confidence in Dependability Arguments", Proceedings DSN 2003, pp. 25-34, IEEE Computer Society, ISBN 0-7695-1952-0, 2003.

[Bloomfield and Littlewood 07] R E Bloomfield, and B Littlewood (2007). "Confidence: its role in dependability cases for risk assessment". International Conference on Dependable Systems and Network, Edinburgh, IEEE Computer Society.

[Bloomfield et al 98] R E Bloomfield, P G Bishop, C C M Jones, P K D Froome, ASCAD—Adelard Safety Case Development Manual, Adelard 1998, ISBN 0-9533771-0-5.

[Bloomfield et al 06] R E Bloomfield, S Guerra, A Miller, M Masera and C B Weinstock, "International Working Group on Assurance Cases (for Security)," IEEE Security and Privacy, **4** (3), pp. 66-68, May/June, 2006.

[Bloomfield *et. al*. 2008] R. Bloomfield, I. Gashi, A. Povyakalo and V. Stankovic, "Comparison of Empirical Data from Two Honeynets and a Distributed Honeypot Network", in *Proceedings of ISSRE-2008, 19th International Symposium on Software Reliability Engineering*, Redmond, Washington, Nov. 2008.

[Bobbio 98] A. Bobbio, A. Puliafito, M. Telek and K. S. Trivedi, "Recent Developments in Non-Markovian Stochastic Petri Nets", Journal of Circuits, Systems, and Computers 8(1): 119-158 (1998)

[Bolstad 2000] Bolstad, C. A. (2000) Age-related factors affecting the perception of essential information during risky driving situations. *Paper presented at the Human Performance Situation Awareness and Automation: User-Centered Design for the New Millennium Conference.* Savannah, GA.

[Bondavalli et al. 00] A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli, and F. Sandrini. DEEM: a tool for the dependability modeling and evaluation of multiple phased systems. In Proc. of the Int. Conference on Dependable Systems and Networks (DSN2000), pages 231-236, New York, USA, June 2000.

[Bondavalli 01] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza and G. Savoia, "Dependability Analysis in the Early Phases of UML Based System Design", Int. Journal of Computer Systems - Science & Engineering, Vol. 16 (5), pp. 265-275, 2001.

[Bondavalli 07a] Bondavalli, A., Ceccarelli, A., Falai, L., and Vadursi, M. 2007. Foundations of Measurement Theory Applied to the Evaluation of Dependability Attributes. In Proceedings of the 37th Annual IEEE/IFIP international Conference on Dependable Systems and Networks (June 25 - 28, 2007). DSN. IEEE Computer Society, Washington, DC, 522-533.

[Bondavalli 07b] A. Bondavalli, A. Ceccarelli, L. Falai, and M. Vadursi., "Towards making Nekostat a proper measurement tool for the validation of distributed systems", in Proceedings of The 8th International Symposium on Autonomous Decentralized Systems, March 2007.

[Bondavalli *et al.* 2009] A. Bondavalli, P. Lollini and L. Montecchi. QoS Perceived by Users of Ubiquitous UMTS: Compositional Models and Thorough Analysis. To appear in Journal of Software, Special issue on Selected Papers of The 6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2009.

[Boring et al 2005] Boring, R. L., Gertman, D. I., Joe, J. C. & Marble, J. L. (2005) Human reliability analysis in the U.S. nuclear power industry: a comparison of atomistic and holistic methods. *Human Factors and Ergonomics Society Annual Meeting.*

[Boudali 07] H. Boudali, P. Crouzen, M. Stoelinga: A Compositional Semantics for Dynamic Fault Trees in Terms of Interactive Markov Chains. ATVA 2007: 441-456

[Bourne et al 81] A J Bourne, G T Edwards, D M Hunns, D R Poulter and I A Watson. Defences against common-mode failures in redundancy systems, UKAEA Safety and Reliability Directorate, 1981.

[Brocklehurst 92] S. Brocklehurst and B. Littlewood, "New Ways to get Accurate Reliability Measures," IEEE Software, vol. 9, no. 4, pp.34-42, 1992.

[Brocklehurst et al. 1994] S Brocklehurst, B Littlewood, T Olovsson, E Jonsson, "On Measurement of Operational Security" Proc. Ninth Ann. IEEE Conf. Computer Assurance, Gaithersburg, 1994 (COMPASS '94), ISBN 07803 -1855-2.

[Brooker 2006] Brooker, P. (2006) Longitudinal collision risk for ATC track systems: A hazardous event model. *Journal of Navigation,* 59**,** 55-70.

[Brosgol 2008] B. M. Brosgol, "Safety and Security: Certification Issues and Technologies", Crosstalk, The Journal of Defense Software Engineering, Oct. 2008.

[Brown 00] A. Brown and D.A. Patterson, "Towards Availability Benchmarks: A Cases Study of Software RAID Systems", in Proceedings of the 2000 USENIX Annual Technical Conference, San Diego, CA, USA, USENIX Association, 2000.

[Brown 01] A. Brown and D.A. Patterson, "To Err is Human", First Workshop on Evaluating and Architecting System Dependability (EASY), Joint organized with IEEE/ACM 28th International Symposium on Computer Architecture (ISCA) and the IEEE International Conference on Dependable Systems and Networks, DSN-2001, Göteborg, Sweden, July 2001.

[Brown 02] A. Brown, L.C. Chung and D.A. Patterson. "Including the Human Factor in Dependability Benchmarks," Workshop on Dependability Benchmarking, in Supplemental Volume of DSN 2002, pp. F-9-14, Washington, D.C., USA, 2002.

[Brown et al 2002] Brown, A., B., Chung, L. & Patterson, D., A. (2002) Including the Human Factor in Dependability Benchmarks. *Proceedings of the DSN Workshop on Dependability Benchmarking*

[Brown 04a] A. Brown, L. Chung, W. Kakes, C. Ling, D. A. Patterson, "Dependability Benchmarking of Human-Assisted Recovery Processes", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2004, Florence, Italy, June 2004.

[Brown 04b] A. B. Brown, J. Hellerstein, M. Hogstrom, T. Lau, S. Lightstone, P. Shum, and M. P. Yost, "Benchmarking Autonomic Capabilities: Promises and Pitfalls", 1st International Conference on Autonomic Computing, ICAC 2004, New York, NY, USA, May 2004.

[Brown et al 2004] Brown, A., B., Chung, L., Kakes, W., Ling, C. & Patterson, D., A. (2004) Experience with Evaluating Human-Assisted Recovery Processes. *Proceedings of the 2004 International Conference on Dependable Systems and Networks.* IEEE Computer Society.

[BSI] https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/assurance/641-BSI.html

[BSI-BritishStandards 2001] "BS EN 50128".

[Buchacker 03] K. Buchacker, M. Dal Cin, H.-J. Hoxer, R. Karch, V. Sieh, and O. Tschache, "Reproducible Dependability Benchmarking Experiments Based on Unambiguous Benchmark Setup Descriptions", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2003, San Francisco, CA, USA, June 22-25, 2003.

[Butler and Finelli 93] R Butler and G Finelli, 1993, "The infeasibility of quantifying the reliability of life-critical real-time software," IEEE Transactions on Software Engineering **19**(1), pp. 3-12.

[CAA 98] CAA CAP 670, "SW01—Regulatory Objectives for Software Safety Assurance", CAP 670 Air Traffic Services Safety Requirements. CAA Safety Regulation Group, 1998.

[CAIDA] CAIDA, *"Home Page of the CAIDA project:* http://www.caida.org".

[Campanile, F., A. Cilardo, et al. 2007]. Adaptable Parsing of Real-Time Data Streams. Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, IEEE Computer Society.

[Carmo et al. 97] R. M. L. R. Carmo, L. R. de Carvalho, E. de Souza e Silva, M. C. Diniz, and R. R. R. Muntz. TangramII: A performability modeling environment tool. In Lecture Notes in Computer Science, number 1245, pages 6-18. R. Marie, B. Plateau, M. Calzarossa, and G. Rubino, 1997.

[Carreira 98] J. Carreira, H. Madeira, and J. G. Silva, "Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers," IEEE Transactions on Software Engineering, vol. 24, pp. 125-136, 1998.

[CC 2006] CC, Common Criteria Portal Home page, available online at http://www.commoncriteriaportal.org/, last visited in May 2006, 2006.

[CCriteria 99] Common Criteria, "Common Criteria for Information Technology Security Evaluation: User Guide", 1999.

[CEC 93] Commission of the European Communities, "Information Technology Security Evaluation Manual (ITSEM)", 1993.

[CFDR 08] The computer failure data repository (CFDR), available from http://cfdr.usenix.org/; Internet, accessed 25 March 2008

[Cfengine 2008]. "Cfengine - a Datacentre Management Platform." accessed Dec. 2008, from http://www.cfengine.com/.

[Chandler, W. W. 1983] "The Installation and Maintenance of Colossus" Annals of the History of Computing, IEEE 5(3): 260-262

[Chandra 00] R. Chandra, R. M. Lefever, M. Cukier, and W. H. Sanders, "Loki: a state-driven fault injector for distributed systems," in Int. Conf. on Dependable Systems and Networks, 2000, pp. 237-242.

[Chandra, T. D. and S. Toueg 1996]. "Unreliable Failure Detectors for Reliable Distributed Systems." Journal of the ACM (JACM) 43(2): 225-267.

[Chauduri 97] S. Chauduri and U. Dayal. “An overview of data warehousing and OLAP technology”. SIGMOD Record, 26(1):65-74, March 1997.

[Chen and Avizienis 77] L. Chen and A. Avizienis, "On the Implementation of N-Version Programming for Software Fault Tolerance during Program Execution," presented at 1st International Computer Software and Applications Conference, COMPSAC 77, New York, 1977.

[Chen 02] D. Chen, D. Selvamuthu, D. Chen, L. Li, R.R. Some, A.P. Nikora and K. Trivedi, “Reliability and Availability Analysis for the JPL Remote Exploration and Experimentation System,” in Proc. Int. Conf. Dependable Systems and Networks, pp. 337-344, June 2002.

[Chen, Y., P. Li, et al. 2006]. Y. Chen, P. Li et al. “Measuring the Dependability of Web Services for Use in e-Science Experiments”. Third International Service Availability Symposium, ISAS 2006, Helsinki, Finland, Springer.

[Chillarege 89] R. Chillarege and N. S. Bowen, “Understanding Large System Failures - A Fault Injection Experiment”, in Proc. 19th Int. Symp. on Fault-Tolerant Computing (FTCS-19), pp 356-363, IEEE CS Press, 1989.

[Chillarege 95] R. Chillarege, "Orthogonal Defect Classification," in Handbook of Software Reliability Engineering, M. Lyu, Ed.: IEEE Computer Society Press, McGraw-Hill, 1995, pp. Chapter 9.

[Chiola et al. 95] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo, “Greatspn 1.7: Graphical editor and analyzer for timed and stochastic Petri nets”. Performance Evaluation, 24(1-2):47-68, November 1995.

[Choi 92] G. S. Choi and R. K. Iyer, "Focus: An Experimental Environment for Fault Sensitivity Analysis," IEEE Transactions on Computers, vol. 41, pp. 1515-1526, 1992.

[Choi 94] H. Choi, V. G. Kulkarni and K. S. Trivedi, “Markov Regenerative Stochastic Petri Nets”, Performance Evaluation, 20(1-3), pages 337-357, 1994.

[Chong 07] H.K. Chong, J.-J. Quisquater, "Faults, Injection Methods, and Fault Attacks," Design & Test of Computers, IEEE , vol.24, no.6, pp.544-545, Nov.-Dec. 2007

[Christey 07] S. Christey, “Unforgivable Vulnerabilities”, Black Hat Briefings 2007

[Christmansson 96] J. Christmansson and R. Chillarege, "Generation of an Error Set That Emulates Software Faults Based on Field Data," in 26th Int. Symp. on Fault Tolerant Computing, 1996, pp. 304-313.

[Christmansson 98] J. Christmansson, M. Hiller, and M. Rimen, "An Experimental Comparison of Fault and Error Injection," in 9th International Symposium on Software Reliability Engineering (ISSRE-9) 1998, pp. 369-378.

[Ciardo 94] G. Ciardo, R. German and C. Lindemann, “A characterization of the stochastic process underlying a stochastic Petri net”, IEEE Transactions on Software Engineering, Volume 20, Issue 7, July 1994 Page(s):506 – 515.

[Ciardo 99] G. Ciardo and A.S. Miner, “Efficient Reachability Set Generation and Storage Using Decision Diagrams,” in Proc. 20th Int. Conf. Application and Theory of Petri Nets, pp. 6-25, 1999.

[Ciardo & Miner 96] G. Ciardo, and A. S. Miner. Smart: Simulation and markovian analyzer for reliability and timing. In IEEE International Computer Performance and Dependability Symposium (IPDS'96), page 60, Urbana-Champaign, IL, USA, September 1996. IEEE Comp. Soc. Press.

[CIS] Center for Internet Security, http://www.cisecurity.org/.

[CISWG 2004] Corporate Information Security Working Group (CISWG). November 17, 2004 (Revised January 10, 2005). Report of the Best Practices and Metrics Teams, Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census. Government Reform Committee, United States House of Representatives. Available at http://www.educause.edu/ir/library/pdf/CSD3661.pdf.

[Cisco 2008]. "QoS Monitoring Tools." accessed Dec. 2008, from http://www.cisco.com/en/US/tech/tk543/tk759/technologies_tech_note09186a0080094bc3.shtml.

[Civera 03] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "New Techniques for Efficiently Assessing Reliability of Socs," Microelectronics Journal, vol. 34, pp. 53-61, 2003.

[Clark 95] J. A. Clark and D. K. Pradhan, "Fault Injection: A Method for Validating Computing-System Dependability," IEEE Computer, pp. 47-56, June 1995.

[CMM 2003] Carnegie Mellon University. June 2003. "Systems Security Engineering Capability Maturity Model (SSE-CMM): Model Description Document, Version 3.0." Available at http://www.ssecmm.org/docs/ssecmmv3final.pdf.

[Coccoli 02] A. Coccoli, P. Urbán and A. Bondavalli, "Performance Analysis of a Consensus Algorithm Combining Stochastic Activity Networks and Measurements", in Proc. Int. Conf. on Dependable Systems and Networks (DSN-2002), pp. 551-560, IEEE CS Press, 2002.

[Corral 2005] G. Corral, A. Zaballos, X. Cadenas, and A. Grane, "A distributed vulnerability detection system for an intranet" in: Proc. 39th Annual International Conference on Security Technology, 2005. CCST '05, pp: 291- 294, 11-14 Oct. 2005.

[Coles et al. 2008] Coles, R., Griffin, J., Johnson, H. et al. "Trust Economics Feasibility Study". In 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, June 24-27, 2008, Anchorage, Alaska, pp A45-A50, IEEE Computer Society, 2008

[Constantinescu05a] "Neutron SER characterization of microprocessors", Proceedings of IEEE Dependable Systems and Networks Conference, pp. 754 – 759, 2005.

[Constantinescu 05b] "Dependability benchmarking using environmental tools", Proceedings of IEEE Annual Reliability and Maintanability Symposium, pp. 567-571, 2005.

[Costa 03] D. Costa, H. Madeira, J. Carreira, J. G. Silva, "Xception™: a Software Implemented Fault Injection Tool", in Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation, Alfredo Benso and Paolo Prineto, October 2003.

[Courtois 01] P J Courtois (2001), Semantic Structures and Logic Properties of Computer-Based System Dependability Cases", Nuclear Engineering and Design, **203**, pp.87-106

[Cristian, F. 1989]. "Probabilistic Clock Synchronization." Distributed Computing 3: 146-158.

[CriticalSW] Critical Software SA, Coimbra – Portugal, http://www.criticalsoftware.com/.

[Crucitti *et al.* 2004] P. Crucitti, V. Latora, M. Marchiori and A. Rapisarda, "Error and Attack Tolerance of Complex Networks", *Physical A*, 340 (1-3), pp.388-394, 2004.

[CRUTIAL 06] CRUTIAL - Critical Utility InfrastructurAL Resilience (Project IST-FP6-027513). http://crutial.cesiricerca.it, 2006.

[Csallner 04] C. Csallner, Y. Smaragdakis: "JCrasher: An automatic robustness tester for Java", Practice & Experience, vol. 34, no. 11, Sep. 2004, pp. 1025-1050

[CST] Core Impact, Core Security Technologies, available at http://www.coresecurity.com

[Cukier et al. 2006] M. Cukier, R. Berthier, S. Panjwani, and S. Tan. "A Statistical Analysis of Attack Data to Separate Attacks". In *Proceedings of the international Conference on Dependable Systems and Networks (DSN),* June 25 - 28, 2006, IEEE Computer Society, Washington, DC, pp. 383-392, 2006.

[Cukier 2007] M. Cukier and A. Sharma. "Password Changes: Empirical Results", *ISAST Transactions on Computers and Software Engineering*, v.1, pp. 11-16, 2007.

[Cullen 90] Lord Cullen, The public inquiry into the piper alpha disaster. HMSO Cm 1310, 1990.

[Curnow 76] H J Curnow, B A Wichmann, “A synthetic benchmark”, Computer Journal, Vol 19, No 1, pp43-49, 1976

[Cymru 2004] Cymru, "Team Cymru: The Darknet Project.http://www.cymru.com/Darknet", 2004.

[Czarnecki 03] K. Czarnecki and S. Helsen, “Classification of Model Transformation Approaches”, in Proc. of OOPSLA'03, Anaheim, California, USA, 2003.

[Dacier *et al.* 1996] M. Dacier, Y. Deswarte and M. Kaâniche, “Models and Tools for Quantitative Assessment of Operational Security”, in *12th International Information Security Conference (IFIP/SEC'96),* (S. K. Katsikas and D. Gritzalis, Eds.), (Samos, Greece), pp.177-186, Chapman & Hall, 1996.

[DACS 08] DACS, Software Reliability Dataset, available from https://www.thedacs.com/databases/sled/swrel.php; Internet; accessed 15 March 2008.

[Dang et al 2008a] Dang, V. N., Bye, A., Lois, E., Forester, J. A. & Braarud, P. Ø. (2008a) Benchmarking HRA Methods Against Simulator Data – Design and Organization of the International HRA Empirical Study. *PSAM9, Proceedings of the 9th International Probabilistic Safety Assessment and Management Conference.* Hong Kong, China.

[Dang et al 2008b] Dang, V. N., Forester, J., Bye, A., Lois, E., Parry, G. W. & Julius, J. A. (2008b) A Comparison of Results From a Pilot Benchmarking Study of HRA Methods – A Comparison of Method Predictions Against the Outcomes Observed in the Simulator Study. *PSAM9, Proceedings of the 9th International Probabilistic Safety Assessment and Management Conference.* Hong Kong, China.

[Daly et al. 00] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders. “Möbius: An extensible tool for performance and dependability modeling”. In 11th International Conference, TOOLS 2000, volume Lecture Notes in Computer Science, pages 332-336, Schaumnurg, IL, 2000. B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith (Eds.).

[Das 06] S. R. Das, S. Mukherjee, E. M. Petriu, M. H. Assaf, M. Sahinoglu, and W.-B. Jone, "An Improved Fault Simulation Approach Based on Verilog With Application to ISCAS Benchmark Circuits," Instrumentation and Measurement Technology Conference (IMTC 2006), pp. 1902-1907, 24-27 April 2006.

[Dasilva 07] A. Dasilva, J.-F. Martinez, L. Lopez, A.-B. Garcia, and Luis Redondo, "Exhaustif: A fault injection tool for distributed heterogeneous embedded systems," in Proceedings of the 2007 Euro American conference on Telematics and information systems, 2007.

[Dawson 95] S. Dawson, F. Jahanian, and T. Mitton, "A software fault injection tool on real-time Mach," in 16th IEEE Real-Time Systems Symposium, 1995, pp. 130-140.

[Dawson 96] S. Dawson, F. Jahanian, and T. Mitton, "Testing of fault-tolerant and real-time distributed systems via protocol fault injection," in Int. Symp. on Fault Tolerant Computing, 1996, pp. 404-414.

[Dawson 97] S. Dawson, F. Jahanian, and T. Mitton, "Experiments on six commercial TCP implementations using a software fault injection tool," Software: Practice and Experience, vol. 27, pp. 1385-1410, 1997.

[DBench] DBench Project, Project funded by the European Community under the “Information Society Technology” Programme (1998-2002), http://www.dbench.org/.

[Deavours 98a] D. D. Deavours and W.H. Sanders, “An Efficient Disk-Based Tool for Solving Large Markov Models”, Performance Evaluation, vol. 33, pp. 67-84, 1998.

[Deavours 98b] D. D. Deavours and W.H. Sanders, “On-the-Fly’ Solution Techniques for Stochastic Petri Nets and Extensions”, IEEE Trans. Software Eng., vol. 24, no. 10, pp. 889-902, Oct. 1998.

[Dekker 2003] Dekker, S. (2003) Illusions of explanation: a critical essay on error classification. *The International Journal of Aviation Psychology,* 13**,** 95-106.

[Delgado, N., A. Quiroz Gates, et al. 2004]. "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools." IEEE Transactions On Software Engineering 30(12).

[Delong 96] T. A. Delong, B. W. Johnson, and J. A. Profeta, III, "A Fault Injection Technique for VHDL Behavioral-Level Models," IEEE Design & Test of Computers, vol. 13, pp. 24-33, 1996.

[Delude 95] J. Delude, R. Vienneau, "Analyzing Quantitative Data Through the Web", Proceedings of the 6th Annual Dual Use Technologies & Applications Conference, June 1995.

[DeMillo 88] R. DeMillo et al, "An Extended Overview of the Mothra Software Testing Envi¬ronment", Proc. ACM SIGSOFT/IEEE 2nd Workshop on Soft. Testing, Verification, and Analysis, pp. 142-151, Jul. 1988.

[Diaz, M., G. Juanole, et al. 1994]. "Observer-A Concept for Formal On-Line Validation of Distributed Systems." IEEE Transactions on Software Engineering 20(12): 900-913.

[Diaz & Preneel 2004] C. Diaz, and B. Preneel, "Reasoning About the Anonymity Provided by Pool Mixes that Generate Dummy Traffic", In: Proc. 6th International workshop on Information Hiding, Toronto ON, CA, Vol. 3220, May 2004, pp 309-325, 2004.

[Distefano 08] S. Distefano and A. Puliafito, "Dependability Evaluation with Dynamic Reliability Block Diagrams and Dynamic Fault Trees", IEEE Transactions on Dependable and Secure Computing, Vol. 5, No. 2, April-June 2008.

[Dixit 09] A. Dixit, R. Heald, and A. Wood, "Trends from Ten Years of Soft Error Experimentation," in 5th IEEE Workshop on Silicon Errors in Logic - Systems Effects (SELSE-5), Stanford, CA, USA, March 24-25, 2009.

[Dobson and Randell 86] J. E. Dobson and B. Randell, "Building Reliable Secure Systems out of Unreliable Insecure Components," presented at Conference on Security and Privacy, Oakland, 1986.

[DoD 85] Department of Defense, "Trusted Computer System Evaluation Criteria", 1985.

[Dougherty 1990] Dougherty, E. M. (1990) Human Reliability-Analysis - Where Shouldst Thou Turn. *Reliability Engineering & System Safety,* 29**,** 283-299.

[DShield] DShield, *"Home page of the DShield.org Distributed Intrusion Detection System:* http://www.dshield.org*. "*

[Duffey 08] R. B. Duffey, "The quantification of resilience: learning environments and managing risk," presented at 3rd Symposium on Resilience Engineering, Antibes, France, 2008.

[Durães 02a] J. Durães and H. Madeira, "Characterization of Operating Systems Behaviour in the Presence of Faulty Drivers Through Software Fault Emulation", in Proceedings of the 2002 Pacific Rim International Symposium on Dependable Computing, PRDC-2002, pp. 201-209, Tsukuba, Japan, December 2002.

[Durães 02b] J. Durães and H. Madeira, "Emulation of Software Faults by Educated Mutations at Machine-Code Level," in Int. Symp. on Software Reliability Engineering, 2002, pp. 329-340.

[Durães 03] J. Durães and H. Madeira, "Definition of Software Fault Emulation Operators: A Field Data Study," in Int. Conf. on Dependable Systems and Networks, 2003, pp. 105-114.

[Durães 04a] J. Durães and H. Madeira, "Generic Faultloads Based on Software Faults for Dependability Benchmarking", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2004, Florence, Italy, June, 2004.

[Durães 04b] J. Durães, M. Vieira, and H. Madeira, "Dependability Benchmarking of Web-Servers", Proceedings of The 23rd International Conference on Computer Safety, Reliability and Security, SAFECOMP 2004, Potsdam, Germany, September, 2004.

[Durães 06] J. Durães, Faultloads Based on Software Faults for Dependaility Benchmarking, PhD Thesis, Department of Information Engineering, University of Coimbra, 2006.

[Duraes 06] J. Duraes, H. Madeira, "Emulation of Software Faults: A Field Data Study and a Practical Approach", IEEE Transactions on Software Engineering, Vol. 32, # 11, pp. 849-867, IEEE, November 2006

[Echtle 92] K. Echtle and M. Leu, "The EFA fault injector for fault-tolerant distributed system testing," in IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, Amherst, MA, USA, 1992, pp. 28-35.

[Echtle 95] K. Echtle and M. Leu, "Test of fault tolerant distributed systems by fault injection," in Fault-Tolerant Parallel and Distributed Systems, D. Pradhan and D. Avresky, Eds.: IEEE Computer Society Press, 1995, pp. 244-251.

[EECDirective 99] EEC Directive 90/C81/01, "Emission Test Cycles for the Certification of light duty vehicles in Europe", EEC Emission Cycles, 1999, available at: http://www.dieselnet.com/standards/cycles.

[Eigenmann 01] Rudolf Eigenmann (Editor), "Performance Evaluation and Benchmarking With Realistic Applications", MIT Press, 2001.

[Ejlali 07] A. Ejlali and G. Miremadi, "Error propagation analysis using FPGA-based SEU-fault injection", Microelectronics Reliability, vol. 48, no. 2, February 2008, pp. 319-328.

[Elling 08] Richard Elling, Ira Pramanick, James Mauro, William Bryson, Dong Tang, "Analytical RAS Benchmarks", in (to apear) White Book on Dependability Benchmarking, Karama Kanoun & Lisa Spainhower Eds., 2008.

[Embrey et al 1984] Embrey, D. E., Humphreys, P., Rosa, E. A., Kirwan, B. & Rea, K. (1984) SLIM-MAUD: An Approach to Assessing Human Error Probabilities Using Structured Expert Judgment. *NUREG/CR-3518. Vols. I & II.* Washington, DC, Brookhaven National Laboratory for the U.S. Nuclear Regulatory Commission.

[Emmet and Cleland 02] L Emmet and G Cleland, Graphical Notations, Narratives and Persuasion: a Pliant Systems Approach to Hypertext Tool Design, in Proceedings of ACM Hypertext 2002 (HT'02), College Park, Maryland, USA, June 11-15, 2002

[EN 50126] CENELEC European Standard EN 50126 "Railway Applications: The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)", 1998.

[EN 50128] CENELEC European Standard EN 50128 "Railway Applications: Communications, Signalling and Processing Systems – Software for railway control and protection systems", 2000.

[EN 50129] CENELEC European Standard EN 50129 "Railway Applications - Communication, signalling and processing systems: Safety related electronic systems for signalling", 2001.

[EN 61508] CENELEC European Standard EN 61508 "Functional safety of electrical/electronic/programmable electronic safety-related systems", 2002.

[EN 50159-1] CENELEC European Standard EN 50159-1 "Railway Applications: Communications, Signalling and Processing Systems – Part 1: Safety-related communication in closed transmission systems", 2001;

[EN 50159-2] CENELEC European Standard EN 50159-2 "Railway Applications: Communications, Signalling and Processing Systems – Part 2: Safety-related communication in open transmission systems", 2001.

[Endsley 1995] Endsley, M. R. (1995) Toward a theory of situation awareness in dynamic systems. *Human Factors* 37**,** 32-64.

[Endsley 2003] Endsley, M. R., Bolté, B. & Jones, D. G. (2003) *Designing for situation awareness: an approach to user-centered design*, Taylor & Francis.

[ESARR] ESARR 6, Software in ATM Systems

[Eurocontrol] Eurocontrol Safety Case Development Manual: http://www.eurocontrol.int/cascade/gallery/content/public/documents/safetycasedevmanual.pd

[Falai et al 2005] L. Falai, A. Bondavalli, F. Di Giandomenico. Quantitative Evaluation of Distributed Algorithms Using the Neko Framework: The NekoStat Extension. LADC 2005: 35-51

[Falai 07] L. Falai, Observing, monitoring and evaluating distributed systems, PhD Thesis, University of Florence, December 2007

[Farmer 2002] D. Farmer, and W. Venema, "Improving the Security of Your Site by Breaking Into it",: Network Security Library :: Unix Security, (white paper) 2002.

[FAU] Friedrich-Alexander Universität, Erlangen-Nürnberg – Germany, http://www.uni-erlangen.de/.

[FCTUC] Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra – Portugal, http://www.fct.uc.pt/.

[Feldman 06] Ronen Feldman and James Sanger. The Text Mining Handbook, Cambridge University Press, USA, 2006.

[Fenton 00] N. E. Fenton and N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," IEEE Transactions on Software Engineering, vol. 26, pp. 797-814, 2000.

[Fenton 94] Fenton, N., "Software measurement: a necessary scientific basis," IEEE Transactions on Software Engineering,, vol.20, no.3, pp.199-206, Mar 1994.

[Fenton 97] N.E.Fenton, S.L.Pfleeger, Software Metrics - A rigorous and practical Approach, 2nd Edition, PWS Publishing Company, 1997.

[Fetzer, C. 2007] "Automatic Collection of Failure Traces". Reliability Analysis of System Failure Data workshop, RAF07. Cambridge, UK.

[Fields et al 1997] Fields, B., Harrison, M. & Wright, P. (1997) THEA: Human error analysis for requirements definition. Technical Report YCS-97-294. York, UK, The University of York, Department of Computer Science.

[Finkbeiner, B. and H. Sipma 2004]. "Checking Finite Traces Using Alternating Automata." Formal Methods in System Design 24(2): 101-127.

[Fitzgerald 1999] Fitzgerald, M. J. (1999) DD21's Fatal Flaw. ***U.S. Naval Institute Proceedings.*** U. S. Naval Institute.

[Fonseca 08] J. Fonseca, M. Vieira, "Mapping Software Faults with Web Security Vulnerabilities", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, Anchorage, Alaska, USA, June 24-27, 2008 (to appear).

[Fonseca 08] J. Fonseca, M. Vieira, H. Madeira, "Online Detection of Malicious Data Access Using DBMS Auditing", 23rd Annual ACM Symposium on Applied Computing (ACM SAC 2008), Fortaleza, Ceará, Brazil, March 2008.

[Forester et al 2006] Forester, J., Kolaczkowski, A., Lois, E. & Kelly, D. (2006) Evaluation of human reliability analysis methods against good practices. NUREG-1842 Final Report. U. S. . U.S. Nuclear Regulatory Commission.

[Forester et al 2008] Forester, J., Dang, V. N., Bye, A., Lois, E., Parry, G. W. & Julius, J. A. (2008) Benchmarking Human Reliability Analysis (HRA) Methods against simulator data - Method for the comparison. *PSAM9, Proceedings of the 9th International Probabilistic Safety Assessment and Management Conference.* Hong Kong, China.

[Formal-Systems-Laboratory 2008]. "Monitoring-Oriented Programming." accessed Oct. 2008, from http://fsl.cs.uiuc.edu/index.php/Monitoring-Oriented_Programming.

[Fowler 05] K. Fowler "Dependability," IEEE Instrumentation & Measurement Magazine, Oct 2005, pp. 55-58.

[Franceschinis et al. 02] G. Franceschinis, M. Gribaudo, M. Iacono, N. Mazzocca, and V. Vittorini. Drawnet++: Model objects to support performance analysis and simulation of complex systems. volume 2324 of Lecture Notes in Computer Science, pages 233-238. Springer Verlag, April 2002.

[Fricks et al. 97] R. Fricks, C. Hirel, S. Wells, and K. Trivedi. The development of an integrated modeling environment. In Proceedings of the World Congress on Systems Simulation (WCSS '97), pages 471-476, Singapore, September 1997.

[Fujita and Hollnagel 2004] Fujita, Y. and Hollnagel, E. (2004) Failures without errors: quantification of context in HRA. *Reliability Engineering & System Safety,* 83**,** 145-151.

[Gaba et al 1995] Gaba, D. M., Howard, S. K. & Small, S. D. (1995) Situation awareness in anesthesiology. *Human Factors,* 37**,** 20-31.

[Ganesh 02] J. P. Ganesh and J. B. Dugan, "Automatic Synthesis of Dynamic Fault Trees from UML System Models", in Proc. of the IEEE Int. Symposium on Software Reliability Engineering, (ISSRE), pp 243-256, 2002.

[Gao, Y., Z. Li, et al. 2006]. "A DoS Resilient Flow-level Intrusion Detection Approach for High-speed Networks". Proceedings of 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06).

[Garbin and Shortle 07] D. A. Garbin and J. F. Shortle, "Measuring Resilience in Network-Based Infrastructures" in *Critical Thinking: Moving from Infrastructure Protection to Infrastructure Resilience*, *CIP Program Discussion Paper Series*: George Mason University, 2007, pp. 73-86.

[Garetto et al. 2003] M. Garetto, W. Gong and D. Towsley, "Modeling Malware Spreading Dynamics", in INFOCOM'2003, (San Francisco, CA, USA), 2003.

[Gartner 08] Gartner Inc., Corporate web site available at http://www.gartner.com/, accessed at 31 March 2008

[Garzia, M., M. Khambatt, et al. 2007] "Assessing End-User Reliability Prior To Product Ship". "Reliability Analysis of System Failure Data". Microsoft Research, Cambridge, UK, March 2007.

[Gashi, I., P. Popov, et al. 2007]. "Fault Tolerance via Diversity for Off-the-Shelf Products: A Study with SQL Database Servers." IEEE Transactions on Dependable and Secure Computing (TDSC) 4(4): 280-294.

[German et al. 95] R. German, C. Kelling, A. Zimmermann, and G. Hommel. Timenet: A toolkit for evaluating nonmarkovian stochastic petri-nets. Performance Evaluation, 24:69-87, 1995.

[Gerth, R., D. Peled, et al. 1995]. "Simple on-the-fly Automatic Verification of Linear Temporal Logic". Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification.

[Gertman et al 1988] Gertman, D. I., Gilmore, W. E. & Ryan, T. G. (1988) NUCLARR and human reliability: data sources and data profile. IN Hagen, E. (Ed.) *Conference Record for 1988 IEEE Fourth Conference on Human Factors and Power Plants.* Monterey, CA, IEEE.

[Ghosh 99] A. Ghosh, M. Schmid: "An Approach to Testing COTS Software for Robustness to Operating System Exceptions and Errors", International Symposium on Software Reliability Engineering (ISSRE99), 1999.

[GMU 07] GMU, "Critical Thinking: Moving from Infrastructure Protection to Infrastructure Resilience," George Mason University School of Law, CIP Program Discussion Paper Series February 2007.

[GNOME 2008]. "FreeBSD GNOME Project: Reporting a Bug." accessed Dec. 2008, from http://www.freebsd.org/gnome/docs/bugging.html.

[Goldberg 80] J. Goldberg, "SIFT: A Provable Fault-Tolerant Computer for Aircraft Flight Control," in *Proceedings Information Processing 80*, 1980, pp. 151-156.

[Google 2008a]. "Google BreakPad." accessed Dec. 2008, from http://code.google.com/p/google-breakpad/.

[Google 2008b]. "Socorro." accessed Dec. 2008, from http://code.google.com/p/socorro/.

[Gordon 05] L. A. Gordon, M. P. Loeb, W. Lucyshyn, R. Richardson, Computer Security Institute. Computer crime and security survey, 2005.

[Gordon 06] L. A. Gordon, M. P. Loeb, W. Lucyshyn, R. Richardson, "Computer crime and security survey", Computer Security Institute, 2006.

[Gorges and Staggers 2008] Gorges, M. and Staggers, N. (2008) Evaluations of physiological monitoring displays: a systematic review. *J Clin Monit Comput,* 22**,** 45-66.

[Gorski 04] J Gorski, "Trust Case—A Case for Trustworthiness of IT Infrastructures", in Proceedings of the NATO Advanced Research Workshop on Cyberspace Security and Defence: Research Issues, Gdansk, Poland, September 6-9 2004.

[Goswami 97] K. Goswami, "Depend: A Simulation-Based Environment for System Level Dependability Analysis," IEEE Transactions on Computers, vol. 46, pp. 60-74, 1997.

[Gönczy 06] L. Gönczy, S. Chiaradonna, F. Di Giandomenico, A. Pataricza, A. Bondavalli and T. Bartha, "Dependability evaluation of web service-based processes", in Proc. of European Performance Engineering Workshop (EPEW 2006), LNCS 4054, pp. 166-180, Springer, 2006.

[Gray 85] J. Gray, "Why do Computers Stop and What can be Done About It?," Tandem TR 85.7, June 1985.

[Gray 86] J. Gray, "Why do computers stop and what can be done about it?," presented at 5th Symposium on Reliability in Distributed Software and Database Systems (SRDSDS-5), Los Angeles, CA, USA, 1986.

[Gray 90] J. Gray, "A Census of Tandem Systems Availability Between 1985 and 1990," in IEEE Transactions on Reliability, vol. 39, 1990, pp. 409-418.

[Gray, J. and D. Siewiorek 1991] "High-Availability Computer Systems," Computer 24(9): 19.

[Greenwell et al 06] W S Greenwell, J C Knight, C M Holloway, and J Pease, A Taxonomy of Fallacies in System Safety Argument, 24th International System Safety Conference, Albuequerque, NM, August 2006.

[GRID] Grid Consortium. ICT Vulnerabilities of Power Systems: A Roadmap for Future Research. Deliverable D11. December 2007.

[Gunneflo 89] U. Gunneflo, J. Karlsson, and J. Torin, "Evaluation of Error Detection Schemes Using Fault Injection by Heavy-Ion Radiation," in 19th International Symposium on Fault-Tolerant Computing (FTCS-19), Chicago, IL, USA, 1989, pp. 340-347.

[Gupta, D., R. Gardner, et al. 2005]. "XenMon: QoS Monitoring and Performance Profiling Tool", HP Research Labs.

[Gupta *et al*. 2003] V. Gupta, V. V. Lam, H. V. Ramasamy, W. H. Sanders and S. Singh, "Dependability and Performance Evaluation of Intrusion Tolerant-Server Architectures", in *First Latin-American Symposium on Dependable Computing (LADC 2003),* (Sao-Paulo, Brazil), pp.81-101, IEEE Computer Society, 2003.

[Guthoff 95] J. Guthoff and V. Sieh, "Combining Software-Implemented and Simulation-Based Fault Injection into a Single Fault Injection Method," in 25th Int. Symp. on Fault-Tolerant Computing, 1995, pp. 196-206.

[Haines *et al*. 2003] J. Haines, D. K. Ryder, L. Tinnel and S. Taylor, "Validation of Sensor Alert Correlators", IEEE Security & Privacy, 2003.

[Hale and Heijer 06] A. Hale and T. Heijer, "Is resilience really necessary? the case of railways," in *Resilience Engineering. Concepts and Precepts*, E. Hollnagel, D. D. Woods, and N. Leveson, Eds. Aldershot, UK: Ashgate, 2006, pp. 125-148.

[Hallbert et al 2004] Hallbert, B., Gertman, D., Lois, E., Marble, J., Blackman, H. & Byers, J. (2004) The use of empirical data sources in HRA. *Reliability Engineering & System Safety,* 83**,** 139-143.

[Hallbert and Kolaczkowski 2007] Hallbert, B. and Kolaczkowski, A. (2007) The Employment of Empirical Data and Bayesian Methods in Human Reliability Analysis: A Feasibility Study. NUREG/CR-6949. Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission.

[Han 01] Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publisher, 2001.

[Han 95] S. Han, K. G. Shin, and H. A.´Rosenberg, "Doctor: An Integrated Software Fault Injection Environment for Distributed Real-Time Systems," in Int. Computer Performance and Dependability Symposium, Erlangen, Germany, 1995, pp. 204-213.

[Hancock and Meshkati 1988] Hancock, P. A. and Meshkati, N. (1988) *Human mental workload*, Elsevier.

[Hand 01] D. Hand, H. Mannila and P. Smyth. Principles of Data Mining. MIT Press, Cambridge, MA, 2001.

[Hannaman et al 1985] Hannaman, G., Spurgin, A. & Lukic, Y. (1985) A Model for Assessing Human Cognitive Reliability in PRA Studies. *IEEE Third Conference on Human Factors and Power Plants.* Monterey, CA, Institute of Electronic and Electrical Engineers.

[Harris et al 1995] Harris, W. C., Hancock, P. A., Arthur, E. J. & Caird, J. K. (1995) Performance, workload, and fatigue changes associated with automation. *International Journal of Aviation Psychology,* 5**,** 169-185.

[Hashempour 05] H. Hashempour, L. Schiano, F. Lombardi, "Evaluation, Analysis, and Enhancement of Error Resilience for Reliable Compression of VLSI Test Data," IEEE Trans. on Instrumentation and Measurement, vol. 54, no. 5, October 2005 pp.1761-1770.

[Haverkort 96] B. Haverkort and I. G. Niemegeers, "Performability Modeling Tools and Techniques", Performance Evaluation, 25(1), pages 17-40, Elsevier, 1996.

[Herrera and Hoyden 08] I. A. Herrera and J. Hovden, "The Leading indicators applied to maintenance in the framework of resilience engineering: A conceptual approach," presented at 3rd Resilience Engineering Symposium, Antibes- Juan Les Pins, France, 2008.

[HewlettPackard 2008]. "HP Software." accessed Dec. 2008, from http://welcome.hp.com/country/us/en/prodserv/software.html.

[HIDENETS 06] HIDENETS - HIghly DEpendable ip-based NETworks and Services (Project IST-FP6-STREP-26979). http://www.hidenets.aau.dk/, 2006.

[Hillston 05]. J. Hillston. Fluid Flow Approximation of PEPA models. In Proceedings of the *Second international Conference on the Quantitative Evaluation of Systems* (September 19 - 22, 2005). QEST. IEEE Computer Society, Washington, DC, 2005.

[Hiltunen , M. A. 1995]. "Membership and System Diagnosis". Proceedings of the 14th Symposium on Reliable Distributed Systems, IEEE Computer Society.

[HMSO 86] Advisory Council on Applied Research and Development, ``Software: A vital key to UK competitiveness'', HMSO 1986.

[Hoarau 05] W. Hoara, S. Tixeuil, "A language-driven tool for fault injection in distributed systems," 6th IEEE/ACM International Workshop on Grid Computing, pp.13-14, 2005

[Hoarau 06] W. Hoarau, S. Tixeuil, F. Vauchelles, "Fault injection in distributed Java applications," 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), pp. 25-29, 2006

[Hogg et al 1995] Hogg, D. N., Folleso, K., Strandvolden, F. & Torralba, B. (1995) Development of a Situation Awareness Measure to Evaluate Advanced Alarm Systems in Nuclear-Power-Plant Control Rooms. *Ergonomics,* 38**,** 2394-2413.

[Hollnagel 1993] Hollnagel, E. (1993) *Human reliability analysis: Context and control,* London, Academic Press.

[Hollnagel 1998] Hollnagel, E. (1998) *Cognitive Reliability and Error Analysis Method – CREAM,* Oxford, Elsevier Science.

[Hollnagel 2002] Hollnagel, E. (2002) Dependability of Joint Human-Computer Systems IN Anderson, S. (Ed.) *Computer Safety, Reliability and Security, SAFECOMP 2002, LNCS 2434.* Springer Berlin / Heidelberg.

[Hollnagel 2005] Hollnagel, E. (2005) Human Reliability Assessment in Context. *Nuclear Engineering and Technology,* 37**,** n/a.

[Hollnagel et al 2006] Hollnagel, E., Woods, D. D. & Leveson, N. (2006) *Resilence Engineering: Concepts and Precepts,* Aldershot, Ashgate.

[HoneynetProject 2008]. "The Honeynet Project." accessed Dec. 2008, from http://www.honeynet.org/.

[Horton *et al.* 98]. G. Horton, V. G. Kulkarni, D. M. Nicol, and K. S. Trivedi. Fluid stochastic Petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, Volume 105, Issue 1, 16 February 1998, Pages 184-201.

[Howell 2008] Howell, W. C. (2008) Human Factors in the 1990s: Sealing transition cracks. *Human Factors,* 50**,** 354-358.

[Hsueh 97] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," Computer, vol. 30, pp. 75-82, 1997.

[Hsueh, M. C. 2008]. "Failure Characterization of a Large Enterprise Computing Environment". Int. Workshop on Resilience Assessment and Dependability Benchmarking (RADB 2008), in DSN2008, Anchorage, Alaska, IEEE Computer Society.

[Hughes and Hall 92] G Hughes and R S Hall, "Recent developments in protection and safety-related systems for Nuclear Electric's (UK) power plant", in IAEA/OECD International Symposium on Nuclear Power Plant Instrumentation and Control, Tokyo, 1992.

[Hunns and Wainwright 91] D M Hunns and N Wainwright (1991), "Software-based protection for Sizewell B: the regulator's perspective", Nuclear Engineering International, September, pp.38-40.

[I3P] I3P Grand Challenge Ideas, December 23, 2005

[IBM 99] IBM. Intelligent Miner for Data, Applications Guide. 1999.

[IBM 2008a]. "Autonomic Computing." accessed Dec. 2008, from http://www.research.ibm.com/autonomic/index.html.

[IBM 2008b]. "Tivoli Software." accessed Dec. 2008, from http://www-01.ibm.com/software/tivoli/.

[ICI 1974] ICI (1974) (Imperial Chemical Industries Ltd.) Hazard and Operability Studies. Process Safety Report 2.

[IEC 93] International Electrotechnical Commission, "Functional safety of electrical/electronic/programmable electronic systems: Generic Aspects. Part 1: General requirements", a draft standard from International Electrotechnical Commission Sub-Committee 65A: System Aspects, Working Group 10, 1993.

[IEC 98] International Electrotechnical Commission, "Functional Safety of Electrical, Electronic, and Programmable Electronic Safety Related Systems", IEC 61508, Parts 1 to 7, 1998 to 2000.

[IEE 89] The Institution of Electrical Engineers and the British Computer Society, ``Software in Safety Related Systems'', IEE October 1989.

[IEEE] IEEE Standard Glossary of Software Engineering Terminology, URL: http://standards.ieee.org/

[IEEE Std 1149.1 01] IEEE Std 1149.1-2001 - IEEE Standard Test Access Port and Boundary-Scan Architecture: IEEE, Piscataway, NJ 08854 USA, 2001.

[IEEE-1802 1997] IEEE-1082 (1997) IEEE Guide for Incorporating Human Action Reliability Analysis for Nuclear Power Generating Stations.

[IEEE-ISTO 5001 03] IEEE-ISTO 5001 - the Nexus 5001 Forum™ Standard for a Global Embedded Processor Debug Interface: IEEE-ISTO, Piscataway, NJ 08854 USA, 2003.

[IETF 1990] "Request for Comments (RFC) 1157" accessed Dec. 2008, from http://tools.ietf.org/html/rfc1157

[IETF 1995] "Request for Comments (RFC) 1757" http://tools.ietf.org/html/rfc1757

[IETF 1997] "Request for Comments (RFC) 2021" http://tools.ietf.org/html/rfc2021

[IETF 2002] "Request for Comments (RFC) 3413" accessed Dec. 2008, from http://www.rfc-editor.org/rfc/rfc3413.txt

[IFIPWG10.4] IFIP WG10.4 on Dependable Computing And Fault Tolerance, http://www.dependability.org/wg10.4/.

[IPSec 2008] "Intrusion Detection and Prevention Tree" accessed Jun. 2009, from http://ipsec.pl/intrusion-detection/prevention-systems-classification-tree.html

[SO/IEC 25051] ISO/ IEC JTC 1 "The SO/IEC 25051 standard is the Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) — Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing"

[Israr 07] T. Israr, M. Woodside and G. Franks, "Interaction Tree Algorithms to Extract Effective Architecture and Layered Performance Models from Traces", Journal of Systems and Software, Vol 80 (4), pp 474-492, 2007.

[ITSEC 1991] ITSEC, Information Technology Security Evaluation Criteria, Office for Official Publications of the European Communities, June 1991.

[Iyer, R. K. 1995]. Experimental Evaluation. Proceedings of the 25th International Symposium on Fault-Tolerant Computing, FTCS-25, Pasadena, California.

[Jackson et al 07] D Jackson, M Thomas, and L I Millett, Editors, "Software for Dependable Systems: Sufficient Evidence?" Committee on Certifiably Dependable Software Systems, National Research Council ISBN: 0-309-10857-8, 2007.

[Jaquith 2007] A. Jaquith, "Security Metrics: Replacing Fear, Uncertainty, and Doubt", Addison-Wesley, 2007.

[Jenn 94] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, "Fault Injection into Vhdl Models: The Mefisto Tool," in 24th Int. Symposium on Fault-Tolerant Computing Pasadena, CA, USA, 1994, pp. 66-75.

[Jiang, Y., C. K. Tham, et al. 2000]. "Challenges and Approaches in Providing QoS Monitoring." International Journal of Network Management 10(6): 323-334.

[Johnson et al 2007] Johnson, T. R., Tang, X., Graham, M. J., Brixey, J., Turley, J. P., Zhang, J., Keselman, A. & Patel, V. L. (2007) Attitudes toward medical device use errors and the prevention of adverse events. *Jt Comm J Qual Patient Saf*, 33**,** 689-94.

[Jones and Endsley 2000] Jones, D. G. and Endsley, M. R. (2000) Examining the validity of real-time probes as a metric of situation awareness. *Proceedings of the 14th Triennial Congress of the*

International Ergonomics Association and the 44th Annual Meeting of the Human Factors and Ergonomics Society. Santa Monica, CA, Human Factors and Ergonomics Society.

[Jones et al 01] C Jones, R E Bloomfield, P K D Froome, and P G Bishop, "Methods for assessing the safety integrity of safety-related software of uncertain pedigree (SOUP)", Health and Safety Executive Contract Research Report CRR 337/2001, HSE, ISBN 0 7176 2011 5, 2001.

[Jonsson & Olovsson 1997] E. Jonsson and T. Olovsson, "A Quantitative Model of the Security Intrusion Process Based on Attacker Behavior" IEEE Transactions on Software Engineering, vol. 23, p.235-245, 1997.

[Jonsson 2006] E. Jonsson, "Towards an integrated conceptual model of security and dependability," in Proceedings of the First International Conference on Availability, Reliability and Security (AReS), 2006.

[Joyce, J., G. Lomow, et al. 1987]. "Monitoring Distributed Systems." ACM Transactions on Computer Systems (TOCS) 5(2): 121-150.

[Kaâniche 06] M. Kaâniche et al., "Methodologies synthesis", EU FP6 IST project CRUTIAL, deliverable D3, http://crutial.cesiricerca.it, Public deliverables section, December 2006.

[Kaaniche *et.al*. 2006] M. Kaaniche, Y. Deswarte, E. Alata, M. Dacier, V. Nicomette, "Empirical analysis and statistical modeling of attack processes based on honeypots", Supplemental volume of the 2006 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2006).

[Kaaniche *et.al*. 2008] M. Kaâniche, P. Lollini, A. Bondavalli, and K. Kanoun. Modeling the Resilience of Large and Evolving Systems. In International Journal of Performability Engineering (editor-in-chief: Dr. Krishna B. Misra), Volume 4, Number 2, pp. 153-168, April, 2008.

[Kaiser, G., P. Gross, et al. 2002]. "An Approach to Autonomizing Legacy Systems." Workshop on Self-Healing, Adaptive and self-MANaged Systems (SHAMAN). New York City, NY, USA.

[Kalakech 04a] A. Kalakech, T. Jarboui, A. Arlat, Y. Crouzet and K. Kanoun. "Benchmarking Operating Systems Dependability: Windows as a Case Study", in Proceedings of the 2004 Pacific Rim International Symposium on Dependable Computing, PRDC 2004, Papeete, Polynesia, 2004.

[Kalakech 04b] A. Kalakech, K. Kanoun, Y. Crouzet and A. Arlat. "Benchmarking the Dependability of Windows NT, 2000 and XP," in Proceedings of the International Conference on Dependable Systems and Networks, DSN 2004, Florence, Italy, 2004.

[Kanawati 92] G. A. Kanawati, N. A. Kanawati, and J. A. Abraham, "Ferrari: A Tool for the Validation of System Dependability Properties," in 22nd Int. Symp. on Fault-Tolerant Computing 1992, pp. 336-344.

[Kanawati 95a] N. A. Kanawati, G. A. Kanawati, and J. A. Abraham, "Dependability Evaluation Using Hybrid Fault/Error Injection," in International Computer Performance and Dependability Symposium (IPDS'95), 1995, pp. 224-233.

[Kanawati 95b] G. Kanawati, N. Kanawati and J. Abraham, "FERRARI: A Flexible Software-Based Fault and Error Injection System", IEEE Trans. Computers, vol. 44, no. 2, February 1995.

[Kanoun 04] K. Kanoun et al., http://www.laas.fr/DBench, Project Reports section, project full final report, 2004.

[Kanoun 05] K. Kanoun, Y. Crouzet, A. Kalakech, A.-E. Rugina and P. Rumeau, "Benchmarking the Dependability of Windows and Linux using Postmark workloads", in Proc. 16th Int. Symposium on Software Reliability Engineering (ISSRE-2006), Chicago, USA, 2005.

[Kanoun 06] K. Kanoun and Y. Crouret, "Dependability Benchmarking for Operating Systems", in International Journal of Performance Engineering, vol. 2, no. 3, pp. 275-287, 2006.

[Kanoun 08] Karama Kanoun (Editor), Lisa Spainhower (Editor), "The White Book on Dependability Benchmarking for Computer Systems", (to apear) 2008.

[Kao 93] W. I. Kao, R. K. Iyer, and D. Tang, "Fine: A Fault Injection and Monitoring Environment for Tracing the Unix System Behavior under Faults" IEEE Transactions on Software Engineering, vol. 19, pp. 1105-1118, 1993.

[Kao 95] W.-l. Kao and R. K. Iyer, "Define: A Distributed Fault Injection and Monitoring Environment," in Fault-Tolerant Parallel and Distributed Systems, D. Pradhan and D. Avresky, Eds.: IEEE Computer Society Press, 1995, pp. 252-259.

[Karlsson 91] J. Karlsson, U. Gunneflo, P. Liden, and J. Torin, "Two Fault Injection Techniques for Test of Fault Handling Mechanisms," in Int. Test Conference, Nashville, TN, USA, 1991, pp. 140-9.

[Katcher 97] J. Katcher, "PostMark: A New File System Benchmark, Network Appliance", www.netapp.com/tech_library/3022.html, N°3022, 1997.

[KDE 2008]. "KDE Bug Tracking System." accessed Dec. 2008, from https://bugs.kde.org/.

[Kellington 07] J. W. Kellington, R. McBeth, P. Sanda, and R. N. Kalla, "IBM Power6 Processor Soft Error Tolerance Analysis Using Proton Irradiation," in 3rd IEEE Workshop on Silicon Errors in Logic - Systems Effects (SELSE-3), Austin, TX, USA, 2007.

[Kelly 98] T P Kelly, Arguing Safety: A Systematic Approach to Managing Safety Cases, PhD thesis, Univ. of York, 1998.

[Kelly and Weaver 04] T P Kelly and R A Weaver, "The Goal Structuring Notation - A Safety Argument Notation", Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004.

[Kemeney 60] J.G. Kemeney and J.L. Snell, "Finite Markov Chains", D. Van Nostrand Company, Inc., 1960.

[Kiczales, G., J. Lamping, et al. 1997] "Aspect-Oriented Programming". Proceedings of the European Conference on Object-Oriented Programming.

[Kimball 02] Ralph Kimball and Margy Ross, "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)", Ed. J. Wiley & Sons, Inc, ISBN: 0471200247, 2002.

[Kirwan 1990] Kirwan, B. (1990) A resources flexible approach to human reliability assessment for PRA. *Safety and Reliability Symposium.* Altrincham, Elsevier Applied Sciences, London, UK.

[Kirwan 1994] Kirwan, B. (1994) *A Guide to Practical Human Reliability Assessment,* London, UK, Taylor and Francis.

[Kirwan 1996] Kirwan, B. (1996) The validation of three human reliability quantification techniques -- THERP, HEART and JHEDI: Part 1 -- technique descriptions and validation issues. *Applied Ergonomics,* 27**,** 359-373.

[Kirwan 1997] Kirwan, B. (1997) The validation of three human reliability quantification techniques -- THERP, HEART and JHEDI: part iii -- Practical aspects of the usage of the techniques. *Applied Ergonomics,* 28**,** 27-39.

[Kirwan et al 1997] Kirwan, B., Basra, G. & TaylorAdams, S. E. (1997) CORE-DATA: A computerised human error database for human reliability support. *Global Perspectives of Human Factors in Power Generation***,** 97-912.

[Kirwan and Gibson 2007] Kirwan, B. and Gibson, H. (2007) CARA: A Human Reliability Assessment Tool for Air Traffic Safety Management — Technical Basis and Preliminary Architecture *The Safety of Systems Proceedings of the Fifteenth Safety-critical Systems Symposium* Bristol, UK.

[Kirwan et al 2008] Kirwan, B., Gibson, W. H. & Hickling, B. (2008) Human error data collection as a precursor to the development of a human reliability assessment capability in air traffic management. *Reliability Engineering & System Safety,* 93**,** 217-233.

[KL 2001] D. L. Kewley, and J. Lowry, “Observations on the Effects of Defense in Depth on Adversary Behaviour in Cyber Warfare“, in Proc. of the 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 5-6 June, 2001.

[Kohno *et. al.* 2004] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, "Analysis of an Electronic Voting System," pp.27, 2004 IEEE Symposium on Security and Privacy, 2004.

[Kolaczkowski et al 2005] Kolaczkowski, A., Forester, J., Lois, E. & Cooper, S. (2005) Good Practices for Implementing Human Reliability Analyses. *NUREG-1792 Final Report.* U.S. Nuclear Regulatory Commission.

[Koopman 00] Philip Koopman, John DeVale: “The Exception Handling Effectiveness of POSIX Operating Systems”, IEEE Transactions on Software Engineering, Vol. 26, No. 9, Sept. 2000.

[Koopman 97] P. J. Koopman, J. Sung, C. Dingman, D. P. Siewiorek and T. Marz, “Comparing Operating Systems using Robustness Benchmarks”, in Proceedings of the 16th International Symposium on Reliable Distributed Systems, SRDS-16, Durham, NC, USA, pp.72-9, 1997.

[Kovacs *et al.* 2008] M. Kovacs, P. Lollini, I. Majzik and A. Bondavalli. An integrated framework for the dependability evaluation of distributed mobile applications. In Proc. of the RISE/EFTS Joint International Workshop on Software Engineering for REsilieNt systEms (SERENE 2008), pages 29-38, Newcastle upon Tyne, UK, November 17-19, 2008.

[Kraemer 2004] S. Kraemer, P. Carayon, and R.Duggan, "Red Team Performance for Improved Computer Security", *Human Factors and Ergonomics Society Annual Meeting Proceedings*, vol. 48, 2004.

[Kumar 04] S. Kumar, V. Marbukh, “On Route Exploration Capabilities of Multi-Path Routing in Variable Topology Ad hoc Networks,” Proc. of IEEE IMTC 2004 - Instrumentation and Measurement Technology Conference, Como, Italy, 18-20 May 2004

[Kwang-Ting 99] C. Kwang-Ting, H. Shi-Yu, and D. Wei-Jin, "Fault Emulation: A New Methodology for Fault Grading," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 18, pp. 1487-1495, 1999.

[LAAS-CNRS] Laboratory for Analysis and Architecture of Systems, Toulouse – France, http://www.laas.fr/.

[Lamport, L. 1978]. "Time, clocks, and the ordering of events in a distributed system." Communications of the ACM 21(7): 558-565.

[Lamport, L. 1987]. “Synchronizing Time Servers”, Digital Equipment Corporation.

[Le Bot et al 1999] Le Bot, P., Cara, F. & Bieder, C. (1999) MERMOS, A second generation HRA method: what it does and doesn’t do. *Proceedings of the international topical meeting on probabilistic safety assessment, PSA’99.* Washington, DC, La Grange Park, ILL, American Nuclear Society.

[Leathley 1997] Leathley, B. A. (1997) HAZOP approach to allocation of function in safety critical systems. *ALLFN’97: Revisiting the Allocation of Function Issue.* IEA Press.

[Lee 93] I. Lee, R. K. Iyer, "Faults, Symptoms and Software Fault Tolerance in the Tandem GUARDIAN90 Operating System", in Proceedings of the 23rd IEEE International Symposium on Fault Tolerant Computing - FTCS'93, Toulouse, France, 1993, pp. 20-29

[Lee 95] I. Lee, R. K. Iyer, "Software Dependability in the Tandem GUARDIAN System," in IEEE Transactions on Software Engineering, vol. 21, 1995, pp. 455-467.

[Lemercier and Cellier 2008] Lemercier, C. and Cellier, J. M. (2008) Attention deficits in car driving: Inattention, distraction and interference. *Travail Humain,* 71**,** 271-296.

[Leurre.com 2008]. "The Leurre.com Honeynet Project." accessed Dec. 2008, from http://www.leurrecom.org/.

[Leva et al 2009] Leva, M. C., De Ambroggi, M., Grippa, D., De Garis, R., Trucco, P. & Strater, O. (2009) Quantitative analysis of ATM safety issues using retrospective accident data: The dynamic risk modelling project. *Safety Science,* 47**,** 250-264.

[Levy, R., J. Nagarajarao, et al. 2003]. "Performance Management for Cluster Based Web Services". IFIP/IEEE Eighth International Symposium on Integrated Network Management, 2003.

[Li, P., Y. Chen, et al. 2006]. "Measuring the Dependability of Web Services for Use in e-Science Experiments". ISAS 2006, Springer-Verlag.

[Li, P. L., M. Ni, et al. 2008]. "Reliability Assessment of Mass-Market Software: Insights from Windows Vista®". 19th International Symposium on Software Reliability Engineering, ISSRE 2008, Seattle, WA, USA.

[Liblit, B. R. 2004]. "Cooperative Bug Isolation". School of Computer Science, University of California Berkeley. PhD.

[Lindemann et al. 99] C. Lindemann, A. Reuys, and A. Thümmler. The dspnexpress 2.000 performance and dependability modeling environment. In Proc. of the 29th Annual Int. Symp. on Fault-Tolerant Computing, pages 228-231, Madison, Wisconsin, USA, June 1999.

[Lightstone 03] S. Lightstone, J. Hellerstein, W. Tetzlaff, P. Janson, E. Lassettre, C. Norton, B. Rajaraman, and L. Spainhower. "Towards Benchmarking Autonomic Computing Maturity." in Proceedings of the First IEEE Conference on Industrial Automatics, INDIN-2003, Banff, Canada, August 2003.

[Lions 96] J. L. Lions, "Report by the Inquiry Board on the Ariane 5 Flight 501 Failure," ESA/CNES, Paris 19 July 1996.

[Littlewood 98] B Littlewood, "The Use of Computers in Safety-Critical Applications, Final Report of the Study Group on the Safety of Operational Computer Systems constituted by the Advisory Committee on the Safety of Nuclear Installations", HSE Books London 1998.

[Littlewood 00] B Littlewood (2000), 'The use of proofs in diversity arguments', IEEE Trans Software Engineering, **26** (10), pp.1022-1023.

[Littlewood and Strigini 93] B Littlewood and L Strigini (1993), "Assessment of ultra-high dependability for software-based systems", Communications of the ACM, **36** (11), pp.69-80

[Littlewood & Strigini 2004] B. Littlewood and L. Strigini, "Redundancy and diversity in security", ESORICS 2004, 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, Springer-Verlag LNCS 3193, 2004, pp. 423-438.

[Littlewood and Wright 07] B Littlewood and D Wright (2007). "The use of multi-legged arguments to increase confidence in safety claims for software-based systems: a study based on a BBN of an idealised example." IEEE Trans Software Engineering 33(5): 347-365.

[Littlewood *et al.* 1993] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid and D. Gollmann, "Towards Operational Measures of Computer Security", *Journal of Computer Security*, 2, pp.211-229, 1993.

[Littlewood *et. al.* 2004] B. Littlewood, L. Strigini. "Redundancy and Diversity in Security", in Proc. ESORICS 2004, 9th European Symposium on Research in Computer Security, Sophia Antipolis, France, Springer-Verlag, LNCS 3193, pp. 423-438, 2004.

[Loft et al 2007] Loft, S., Sanderson, P., Neal, A. & Mooij, M. (2007) Modeling and predicting mental workload in en route air traffic control: Critical review and broader implications. *Human Factors,* 49**,** 376-399.

[Lollini 07] P. Lollini, A. Bondavalli et al., "Evaluation methodologies, techniques and tools (final version)", EU FP6 IST project HIDENETS, deliverable D4.1.2, http://www.hidenets.aau.dk/, Public deliverables section, December 2007.

[Lollini *et al.* 2009] P. Lollini, A. Bondavalli and F. di Giandomenico, "A Decomposition-Based Modeling Framework for Complex Systems," Reliability, IEEE Transactions on, vol.58, no.1, pp.20-33, March 2009.

[Looker 03] N. Looker, J. Xu, "Assessing the dependability of OGSA middleware by fault injection", 22nd International Symposium on Reliable Distributed Systems, pp. 293-302, 2003.

[Looker 05a] N. Looker, M. Munro, and J. Xie, "Simulating errors in web services," International Journal of Simulation: Systems, Science and Technology vol. 5, pp. 29-37, 2005.

[Looker 05b] N. Looker, M. Munro, and J. Xie, "A comparison of network level fault injection with code insertion," in 29th Int. Computer Software and Applications Conference, Edinburgh, Scotland 2005.

[Looker 08] N. Looker, M. Munro, J. Xu, "A Tool for Dependability Analysis of Web Services", available at http://www.wsfit.org/, last accessed at 30 March 2008.

[Ludwig, H., A. Keller, et al. 2003]. Web Service Level Agreement (WSLA) - Language Specification, IBM T.J. Watson Research Center.

[Lye & Wing 2005] K.-W. Lye and J. M. Wing, "Game strategies in network security", *International Journal on Information Security*, 4 (1-2), pp.71-86, 2005.

[Lyons 2004] Lyons, M., Adams, S., Woloshynowych, M. & Vincent, C. (2004) Human reliability analysis in healthcare: a review of techniques. *The International Journal of Risk and Safety in Medicine,* 4**,** 223-238.

[Madeira 94] H. Madeira, M. Rela, F. Moreira, and J. G. Silva, "A General Purpose Pin-Level Fault Injector," in European Dependable Computing Conference, Berlin, Germany, 1994, pp. 199-216.

[Madeira 02] H. Madeira, R. Some, F. Moreira, D. Costa, D. Rennels, "Experimental evaluation of a COTS system for space applications", The International Conference on Dependable Systems and Networks, DSN-2002, Bethesda, Maryland, USA, June 2002.

[Madan *et al.* 2002] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan and K. Trivedi, "Modeling and Quantification of Security Attributes of Software Systems", in *IEEE International Conference on Dependable Systems and Networks (DSN 2002),* Washington, DC, USA, pp.505-514, IEEE computer Society, 2002.

[Madeira 03] Madeira, H. and Costa, J. and Vieira, M. , "The OLAP and Data Warehousing Approaches for Analysis and Sharing of Results from Dependability Evaluation Experiments", International Conference on Dependable Systems and Networks, DSN-DCC 2003, San Francisco, CA, USA, June 2003.

[Madeira and Koopman 05] H. Madeira and P. Koopman, "Dependability Benchmarking: making choices in an n-dimensional problem space," 2005.

[Maia 05] R. Maia, L. Henriques, R. Barbosa, D. Costa, H. Madeira, "Xception fault injection and robustness testing framework: a case-study of testing RTEMS", VI Test and Fault Tolerance Workshop (jointly organized with the 23rd Brazilian Symposium on Computer Networks (SBRC)), May 2005.

[Mainkar 96] V. Mainkar and K. Trivedi, "Sufficient Conditions for Existence of a Fixed Point in Stochastic Reward Net-Based Iterative Models", IEEE Trans. Software Eng., vol. 22, no. 9, pp. 640-653, Sept. 1996.

[Mani-Chandy, K. and L. Lamport 1985]. "Distributed Snapshots: Determining Global States of Distributed Systems." ACM Transactions on Computer Systems (TOCS) 3(1): 63-75.

[Mansouri-Samani, M. and M. Sloman 1992]. "Monitoring Distributed Systems (A Survey)". Imperial College Research. London, Imperial College, London**:** 49.

[Marsden 02] E. Marsden, J.-C. Fabre, J. Arlat, "Dependability of CORBA systems: service characterization by fault injection," 21st IEEE Symposium on Reliable Distributed Systems (SRDS 2002), pp. 276-285, 2002

[Martin 07] E. Martin, S. Basu, T. Xie, "WebSob: A Tool for Robustness Testing of Web Services", Proceedings of the 29th International Conference on Software Engineering - Companion, ICSE 2007 Companion, May 2007

[Mauro 04] J. Mauro, J. Zhu and I. Pramanick. "The System Recovery Benchmark," in Proceedings of the 2004 Pacific Rim International Symposium on Dependable Computing, PRDC 2004, Papeete, Polynesia, 2004.

[Maxion 00] Roy A. Maxion and Kymie M. C. Tan, "Benchmarking Anomaly-Based Detection Systems", In Proceedings of the International Conference on Dependable Systems and Networks, pages 623-630, New York, NY, June 25-28, 2000.

[Maxion and Reeder 2005] Maxion, R. A. and Reeder, R. W. (2005) Improving user-interface dependability through mitigation of human error. *Int. J. Hum.-Comput. Stud.*, 63**,** 25-50.

[McCarthy 07] J. A. McCarthy, "Introduction: From Protection to Resilience: Injecting "Moxie" into the Infrastructure Security Continuum," in *Critical Thinking: Moving from Infrastructure Protection to Infrastructure Resilience*, *CIP Program Discussion Paper Series*: George Mason University, 2007, pp. 1-8.

[McDermid 94] J.A. McDermid (1994), "Support for safety cases and safety argument using SAM", Reliability Engineering and Safety Systems, 43(2), pp. 111-127.

[McDermott 1996] McDermott, R. E., Mikulak, R. J. & Beauregard, M. R. (1996) *The Basics of FMEA,* Portland, Productivity Inc

[McLarenElectronicSystems 2008]. "Atlas—Advanced Telemetry Linked Acquisition System." accessed Dec. 2008, from http://www.mclarenelectronics.com/Products/All/sw_atlas.asp.

[McLaughlin, L. 2004]. "Automated Bug Tracking: The Promise and the Pitfalls." IEEE Software 21(1): 100-103.

[Memon 07] Atif M. Memon: "An event-flow model of GUI-based applications for testing", Software Testing, Verification and Reliability, vol. 17, no. 3, pp. 137-157, 2007

[Meyer 2006] R. Meyer, and M. Cukier, "Assessing the Attack Threat due to IRC Channels", in *Proceedings of the international Conference on Dependable Systems and Networks* (DSN), June 25 - 28, 2006, IEEE Computer Society, Washington, DC, USA, pp. 467-472, 2006.

[Meyer et al 2007] Meyer, P., Le Bot, P. & Pesme, H. (2007) MERMOS: an extended second generation HRA method. *2007 Ieee 8th Human Factors and Power Plants and Hprct 13th Annual Meeting***,** 276-283.

[Michie 94] Donald Michie, D. J. Spiegelhalter, C. C. Taylor and John Campbell. Machine Learning, Neural and Statistical Classification. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[Micrium 2008]. "Micrium - Empowering Embedded Systems." accessed Dec. 2008, from http://www.micrium.com/.

[MicroDigital 2008]. "Web Network Management Protocol." accessed Dec. 2008, from http://www.smxrtos.com/pr/barracuda_wnmp.htm.

[Microsoft 2000]. "Windows Management Instrumentation: Background and Overview." accessed Oct. 2008, from http://msdn.microsoft.com/en-us/library/ms811553.aspx.

[Microsoft 2008]. "Introducing Windows Error Reporting." accessed Dec. 2008, from http://msdn.microsoft.com/en-us/isv/bb190483.aspx.

[Micskei 07] Z. Micskei, I. Majzik, F. Tam: “Comparing Robustness of AIS-Based Middleware Implementations”, In Proc. of Int. Service Availability Symposium (ISAS 2007), Durham, New Hampshire, USA, May 21-22, Springer LNCS 4526, pp 20-30, 2007.

[Miguel and Wright 2003] Miguel, A. and Wright, P. (2003) CHLOE: A technique for analysing collaborative systems. *Proceedings of the ninth European conference on cognitive science approaches to process control (CSAPC’03).* Amsterdam, The Netherlands.

[Miller 95] B. Miller et al.: “Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services”, Computer Sciences Technical Report #1268, University of Wisconsin-Madison, April 1995.

[Miremadi 95] G. Miremadi and J. Torin, "Evaluating Processor-Behavior and Three Error-Detection Mechanisms Using Physical Fault-Injection," IEEE Transactions on Reliability, vol. 44, pp. 441-54, 1995.

[Mosier et al 1998] Mosier, K. L., Skitka, L. J., Heers, S. & Burdick, M. (1998) Automation bias: Decision making and performance in high-tech cockpits. *International Journal of Aviation Psychology,* 8**,** 47-63.

[Moraes 07] R. Moraes, J. Durães, R. Barbosa, E. Martins, H. Madeira, “Experimental Risk Assessment and Comparison Using Software Fault Injection”, in Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Dependable Computing and Communications Symposium (DCCS), Edinburgh, UK, June 2007.

[Moray 2003] Moray, N. (2003) Monitoring, complacency, scepticism and eutactic behaviour. *International Journal of Industrial Ergonomics,* 31**,** 175-178.

[Moray 2008] Moray, N. (2008) The good, the bad, and the future: On the archaeology of ergonomics. *Human Factors,* 50**,** 411-417.

[Moreira 03] F. Moreira, R. Maia, D. Costa, N. Duro, P. Rodríguez-Dapena and K. Hjortnaes, “Static and Dynamic Verification of Critical Software for Space Applications”. Proceedings of the Data Systems In Aerospace, DASIA 2003, June 2003.

[Mosleh and Chang 2004] Mosleh, A. and Chang, Y. H. (2004) Model-based human reliability analysis: prospects and requirements. *Reliability Engineering & System Safety,* 83**,** 241-253.

[Munawar, M. and P. Ward 2006]. Adaptive Monitoring in Enterprise Software Systems. In SIGMETRICS 2006 Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML).

[Murphy, B. 2004]. "Automating Software Failure Reporting." ACM Queue 2(8).

[Musa 80] J. D. Musa, “Software Reliability Data”, Data & Analysis Center for Software, January, 1980.

[Narasimhan 08] Priya Narasimhan, “Vajra: Benchmarking Survivability in Distributed Systems”, http://www.cylab.cmu.edu/default.aspx?id=1990, 2008.

[NASA 08] NASA/WVU IV&V Facility, Metrics Data Program, available from http://mdp.ivv.nasa.gov; Internet; accessed 15 March 2008.

[Nelli 96] M. Nelli, A. Bondavalli and L. Simoncini, “Dependability Modeling and Analysis of Complex Control Systems: an Application to Railway Interlocking”, EDCC-2 European Dependable Computing Conference, Lecture Notes in Computer Science N. 1150. Taormina, Italy, Springer-Verlag: 93-110, 1996.

[Nelson 78] R. Nelson, “Software Data Collection and Analysis”, Rome Air Development Center, Rome, NY, September 1978.

[Nemeth 08] C. P. Nemeth, "Resilience Engineering: The Birth of a Notion," in Resilience Engineering Perspectives Volume 1: Remaining Sensitive to the Possibility of Failure, Resilience Engineering Perspectives, E. Hollnagel, C. P. Nemeth, and S. Dekker, Eds.: Ashgate, 2008, pp. 346.

[Nemeth and Cook] C. Nemeth and R. Cook, "Reliability Versus Resilience: What Does Healthcare Need?."

[Neto 08] Afonso Araújo Neto, Marco Vieira, "Towards Assessing the Security of DBMS Configurations", IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, Anchorage, Alaska, USA, June 24-27, 2008.

[Neves et al. 2006] N. Neves, J. Antunes, M. Correia, P. Verissimo, R. Neves. "Using Attack Injection to Discover New Vulnerabilities", Dependable Systems and Networks, Page(s):457 – 466, 2006

[NeXpose] NeXpose, Rapid7, available at http://www.rapid7.com/services/pentest.jsp

[Ng 96] W. T. Ng, C. M.Aycock, G. Rajamani, and P. M. Chen, "Comparing Disk and Memory's Resistance to Operating System Crashes," in 7th Int. Symp. on Software Reliability Engineering, 1996, pp. 185-194.

[Ng 99] W. T. Ng, W. T. Ng, and P. M. Chen, "The Systematic Improvement of Fault Tolerance in the Rio File Cache," in 29th Int. Symp. on Fault-Tolerant Computing, Madison, WI, USA, 1999, pp. 76-83.

[Nichols 2007] E.E. Nichols, and G. Peterson, "A Metrics Framework to Drive Application Security Improvement", Building Security In, IEEE Security & Privacy, pp. 88-91, 2007.

[Nicol 04] D. M. Nicol, W. H. Sanders and K. S. Trivedi, "Model-based Evaluation: From Dependability to Security", IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, pp 48-65, 2004.

[Norman 1988] Norman, D. A. (1988) *The Design of Everyday Things,* New York, NY, Doubleday.

[NIST 08] Error, Fault, and Failure Data Collection and Analysis, available from http://hissa.nist.gov/project/eff.html; Internet; accessed 15 March 2008.

[NTA 07] NTA, May, 2007, http://www.nta-monitor.com/posts/2007/05/annualsecurityreport.html

[Obal & Sanders 98] W.D. Obal, II; W.H. Sanders, "State-space support for path-based reward variables," Computer Performance and Dependability Symposium, 1998. IPDS '98. Proceedings. IEEE International , vol., no., pp.228-237, 7-9 Sep 1998

[ODP] Open Directory Project, http://www.dmoz.org/Computers/Security/Honeypots and Honeynets.

[Olah 2009] J. Olah and I. Majzik, "A Model Based Framework for Specifying and Executing Fault Injection Experiments", In Proc. Fourth International Conference on Dependability of Computer Systems, June 30 – July 2, Brunow Palace, Poland, 2009 (in Press).

[Ortalo *et al.* 1999] R. Ortalo, Y. Deswarte and M. Kaâniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security", *IEEE Transactions on Software Engineering*, 25 (5), pp.633-650, 1999.

[OSSEC 2008]. "OSSEC - a Host-Based Intrusion Detection Stystem." accessed Dec. 2008, from http://www.ossec.net/.

[OWASP] OWASP WSFuzzer Project, available at http://www.owasp.org

[Parasuraman and Riley 1997] Parasuraman, R. and Riley, V. (1997) Humans and automation: Use, misuse, disuse, abuse. *Hum Factors,* 39**,** 230-253.

[Parkin et al. 2008] Parkin, S.E., Yassin Kassab, R. and van Moorsel, A. "The Impact of Unavailability on the Effectiveness of Enterprise Information Security Technologies". In Service Availability. 5th International Service Availability Symposium, ISAS 2008, Tokyo, Japan, May 19-21, 2008. Nanya, T., Maruyama, F., Pataricza, A. et al.(eds.), Lecture Notes in Computer Science, 5017, pp 43-58, Springer, 2008.

[Parry 1995] Parry, G. W. (1995) Suggestions for an Improved Hra Method for Use in Probabilistic Safety Assessment. *Reliability Engineering & System Safety,* 49**,** 1-12.

[Parry et al 2008] Parry, G. W., Lois, E., Forester, J., Dang, V. N., Julius, J. A. & Bye, A. (2008) Insights from Comparison of HRA Method Predictions with Empirical Data on Human Performance in Accident Scenarios. *PSAM9, Proceedings of the 9th International Probabilistic Safety Assessment and Management Conference.* Hong Kong, China.

[Parsons, T, A. Mos, et al. 2008]. "Extracting Interactions in Component-Based Systems." IEEE Transactions on Software Engineering 34(6): 783-799.

[Payne 2006] S. Payne, "A Guide to Security Metrics", SANS Security Essentials Practical Assignment, 2006.

[Penny et al 01] J Penny, A Eaton, PG Bishop and RE Bloomfield. "The Practicalities of Goal-Based Safety Regulation**"**, in Aspects of Safety Management: Proceedings of the Ninth Safety-Critical Systems Symposium Bristol, UK, 6-8 February 2001, Felix Redmill and Tom Anderson (eds.) London; New York: Springer, 2001 ISBN: 1-85233-411-8, pages 35-48.

[Petroski 92] H. Petroski, *To Engineer is Human: The Role of Failure in Successful Design*. New York: St. Martin's Press, 1992.

[Pintér 05] Gergely Pintér, Henrique Madeira, Marco Vieira, András Pataricza and István Majzik. A Data Mining Approach to Identify Key Factors in Dependability Experiments. In Mario Dal Cin and Mohamed Kaâniche and András Pataricza (Eds.) Dependable Computing – EDCC-5, Proceedings of the Fifth European Dependable Computing Conference, Springer, 2005.

[Pintér 08] Gergely Pintér, Henrique Madeira, Marco Vieira, István Majzik and András Pataricza. Integration of OLAP and Data Mining for Analysis of Results from Dependability Evaluation Experiments. In International Journal of Knowledge Management Studies, 2008.

[Pocock et al 2001] Pocock, S., Harrison, M., Wright, P. & Johnson, P. (2001) THEA - a technique for human error assessment early in design. IN Hiros, M. (Ed.) *Human-Computer Interaction: INTERACT'01*. IOS Press.

[Pooley 91] R. J. Pooley. The integrated modeling support environment: A new generation of performance modeling tools. In Proc. of the 5th International Conference in Computer Performance Evaluation: Modeling Techniques and Tools, pages 1-15, Torino, Italy, February 1991. G. Balbo and G. Serazzi.

[Popov et al 00] P. Popov, et al., "Diversity for off-the-Shelf Components," presented at DSN 2000, International Conference on Dependable Systems and Networks - Fast Abstracts supplement, New York, NY, USA, 2000.

[Porcarelli, S., A. Bondavalli, et al. 2002]. "Model-Based Dynamic Reconfiguration in Complex Critical Systems". Fourth European Dependable Computing Conference, (EDCC-4), Fast Abstract Track, Toulouse, France.

[Porcarelli, S., M. Castaldi, et al. 2004]. A Framework for Reconfiguration-based Fault-Tolerance in Distributed Systems Architecting Dependable Systems II. R. De Lemos, C. Gacek and A. Romanovsky, Springer-Verlag.

[Prelude 2008]. "Prelude - a Universal Security Information Management (SIM) System." accessed Dec. 2008, from http://www.prelude-ids.com/en/welcome/index.html.

[PUS 03] Space Engineering - Ground systems and operations - Telemetry and telecommand packet utilization. ECSS-E-70-41A, 30 January 2003.

[Qin, F., J. Tucek, et al. 2005] "RX: Treating Bugs as Allergies - a Safe Method to Survive Software Failures". Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05), New York, NY, USA, ACM Press.

[Raimondi, F., J. Skene, et al. 2008]. Efficient Online Monitoring of Web-Service SLAs. Proceeding of 16th ACM SIGSOFT International Symposium on the Foundations of Software Engineering, FSE 16, Atlanta, Georgia, USA.

[Rajabzadeh 04] A. Rajabzadeh, S. G. Miremadi, and M. Mohandespour, "Experimental Evaluation of Master/Checker Architecture Using Power Supply and Software-Based Fault Injection," in 10th IEEE Int. On-Line Testing Symposium, Funchal, Madeira Island, Portugal, 2004, pp. 239-44.

[Ramachandran 08] Ramachandran, P.; Kudva, P.; Kellington, J.; Schumann, J.; Sanda, P., "Statistical Fault Injection," 38th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008), pp.122-127, 2008.

[Ramirez-Silva & Dacier 2007] E. Ramirez-Silva, and M. Dacier, "Empirical Study of the Impact of Metasploit-Related Attacks in 4 Years of Attack Traces", in: Proc. 12th Asian Computing Science Conference, Doha, Qatar, Springer LNCS 4846, pp. 198-211, December 9-11, 2007.

[Reason 1999] Reason, J. (1999) *Human Error,* Cambridge, UK, Cambridge University Press.

[Rebaudengo 99] M. Rebaudengo and M. Sonza Reorda, "Evaluating the Fault Tolerance Capabilities of Embedded Systems Via Bdm," in 17th IEEE VLSI Test Symposium 1999, pp. 452-457.

[Redmill] F Redmill (eds), Dependability of Critical Computer Systems - 1 and 2, Elsevier Applied Science, ISBN 1 85166-203-0 and ISBN 1 85166-389-1

[RedTeam] Sandia National Laboratories, "Information Operations Red Team and Assessments™", http://www.sandia.gov/iorta/.

[Reer 2008a] Reer, B. (2008a) Review of advances in human reliability analysis of errors of commission - Part 2: EOC quantification. *Reliability Engineering & System Safety,* 93**,** 1105-1122.

[Reer 2008b] Reer, B. (2008b) Review of advances in human reliability analysis of errors of commission, Part 1: EOC identification. *Reliability Engineering & System Safety,* 93**,** 1091-1104.

[Reibman 91]A. Reibman, and M. Veeraraghavan, "Reliability Modeling: An Overview for System Designers", IEEE Computer, April, pp.49-57, 1991.

[Relex™ 2008]. "FRACAS." 2008, from http://www.relex.com/.

[Repository] http://dependability.cs.virginia.edu/info/Safety_Cases:Repository

[ReSIST 06a] ReSIST – Resilience for Survivability in IST (Project IST-0265764). http://www.resist-noe.org/index.html, 2006.

[ReSIST 06b] ReSIST NoE. Resilience-Building Technologies: State of Knowledge, Deliverable D12, URL: http://www.resist-noe.org/outcomes/outcomes.html

[ReSIST 07] ReSIST NoE, Deliverable D13: From Resilience-Building to Resilience-Scaling Technologies: Directions, September 2007.

[RESIST 09] ReSIST, "Deliverable D39: Selected Current Practices," 2009.

[Riha *et al.* 2000] D. S. Riha, B. H. Thacker, H. R. Millwater, Y.-T. Wu, and M. P. Enright, "Probabilistic Engineering Analysis using the NESSUS Software", American Institute of Aeronautics and Astronautics, Southwest Research Institute, 2000.

[Robens 72] Lord Robens, Safety and Health at Work. Report of the Committee 1970 - 72. HMSO Cmnd 5034, 1972.

[Rochlin et al 87] G. I. Rochlin, et al., "The self-designing high-reliability organization: Aircraft carrier flight operations at sea," *Naval War College Review*, vol. 40, pp. 76-90, 1987.

[Rodríguez 99] M. Rodríguez, F. Salles, J.-C. Fabre and J. Arlat, "MAFALDA: Microkernel Assessment by fault injection and design aid", in 3rd European Dependable Computing Conference, EDCC-3, September 1999.

[Rodriguez 02] M. Rodriguez, A. Albinet, J. Arlat, "MAFALDA-RT: a tool for dependability assessment of real-time systems", Proceedings of the International Conference on Dependable Systems and Networks, 2002. DSN 2002, pp.267-272, Bethseda 2002.

[Rosu, G. and K. Havelund 2001]. “Monitoring Java Programs with Java PathExplorer”. Proceedings of the First Workshop on Runtime Verification (RV’01), Paris, France Elsevier.

[Rugina 07] A. E. Rugina, K. Kanoun and M. Kaâniche, “A System Dependability Modeling Framework using AADL and GSPNs”, in Architecting Dependable Systems IV, (R. d. L. e. al., Ed.) vol. LNCS 4615, pp. 14-38, Springer-Verlag, 2007.

[Ruiz 04] J. –C. Ruiz, P. Yuste, P. Gil, and L. Lemus, “On Benchmarking the Dependability of Automotive Engine Control Applications”, IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2004, Florence, Italy, June 2004.

[RVW 2008]. "Runtime Verification Workshop." 2008, from http://www.runtime-verification.org/.

[SAFEDMI 06] SAFEDMI – Safe Driver Machine Interface (DMI) for ERTMS trains (Project IST-FP6-STREP-031413). http://www.safedmi.org/, 2006.

[SAFEDMI 07] SAFEDMI – “Quantitative Evaluation Methodology (Preliminary version)”, D4.1, August 2007.

[Sahner 96] R.A. Sahner, K.S. Trivedi and A. Puliafito, “Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package”, Kluwer Academic Publishers, 1996.

[Sahai, A., V. Machiraju, et al. 2002] “Automated SLA Monitoring for Web Services”. Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Management Technologies for E-Commerce and E-Business Applications, Springer-Verlag.

[Sahoo, R. K., M. S. Squillante, et al. 2004]. “Failure Data Analysis of a Large-Scale Heterogeneous Server Environment”. Proceedings of the Int. Conference on Dependable Systems and Networks (DSN’04), Florence, Italy, IEEE Computer Society.

[Salfner, F., M. Lenk, et al. In print] "A Survey of Online Failure Prediction Methods" ACM Computing Surveys

[Sallhammar et al. 2006] K. Sallhammar, B. E. Helvik and S. J. Knapskog, "A Game-theoretic Approach to Stochastic Security and Dependability Evaluation", in 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC'06), (Indianapolis, USA), 2006.

[Sanders et al. 95] W. H. Sanders, W. D. Obal II, M. A. Qureshi, F. K. Widjanarko: The UltraSAN Modeling Environment. Perform. Eval. 24(1-2): 89-115,1995.

[Sanders 99] W. H. Sanders, “Integrated frameworks for multi-level and multi-formalism modeling”, in Proc. the 8th International Workshop on Petri Nets and Performance Models, pages 2-9, September 1999.

[Sanders 01] W. H. Sanders and J. F. Meyer, “Stochastic activity networks: Formal definitions and concepts”, in Lecture Notes in Computer Science, pages 315-343. Springer-Verlag, 2001.

[Sanders et al. 2006] W. H. Sanders et al., “Measuring Critical Infrastructure Security”, 2006.

[Scarfone, K. and P. Mell 2007]. “Guide to Intrusion Detection and Prevention Systems (IDPS)”, National Institute of Standards and Technology (NIST).

[Schmidt 06] D. C. Schmidt, ”Model-Driven Engineering”, IEEE Computer 39 (2), February 2006.

[Schroeder, B. A. 1995]. "On-line Monitoring: a Tutorial." Computer 28(6): 72-78.

[Schroeder 07a] B. Schroeder, G. A. Gibson, “The Computer Failure Data Repository (CFDR)”, Workshop on Reliability Analysis of System Failure Data (RAF'07), MSR Cambridge, UK, March 2007.

[Schroeder 07b] B. Schroeder, G. A. Gibson, “Disk failures in the real world: What does an MTTF of 1,000,000 hours mean too you?”, 5th Usenix Conference on File and Storage Technologies (FAST’07), 2007.

[Sheridan 2008] Sheridan, T. B. (2008) Risk, human error, and system resilience: Fundamental ideas. *Human Factors,* 50**,** 418-426.

[Shorrock and Kirwan 2002] Shorrock, S. T. and Kirwan, B. (2002) Development and application of a human error identification tool for air traffic control. *Applied Ergonomics,* 33**,** 319-336.

[Shostack & Stewart 2008] A. Shostack and A. Stewart, "The New School of Information Security", Addison Wesley Professional, 2008

[Siegel et al 1975] Siegel, A. I., Wolf, J. J. & Lautman, M. R. (1975) Family of Models for Measuring Human Reliability. *Proceedings Annual Reliability and Maintainability Symposium***,** 110-115.

[Siewiorek, D. and R. Swarz 1998] "General-purpose Computing". Reliable Computer Systems: Design and Evaluation. Natick, Massachusets, A K Peters.

[SIGDeB] IFIP WG 10.4 Dependability Benchmarking SIG, http://www.laas.fr/~kanoun/ifip_wg_10_4_sigdeb/index.html.

[Singh et al 1993] Singh, I. L., Molloy, R. & Parasuraman, R. (1993) Automation-induced "complacency": development of the complacency-potential rating scale. *International Journal of Aviation Psychology,* 3**,** 111-122.

[Sizewell 82] Sizewell B Preconstruction Safety Report, April 1982.

[Skitka et al 1999] Skitka, L. J., Mosier, K. & Burdick, M. D. (1999) Does automation bias decision making? *International Journal of Human-Computer Studies,* 51**,** 991-1006.

[Smith and Harrison 2002] Smith, S. P. and Harrison, M. D. (2002) Blending Descriptive and Numeric Analysis in Human Reliability Design *Interactive Systems:Design, Specification, and Verification.* Berlin / Heidelberg, Springer

[Smith and Stockham 07] P R Smith and R Stockham, EMPHASIS - An Assessment tool for Smart Instruments, PRfsS/Moore Industries-Europe, United Kingdom, 2007.

[Snort.org 2008]. "SNORT® Network Intrusion Prevention and Detection System." accessed Dec. 2008, from http://www.snort.org/.

[Sollenberger and Stein 1995] Sollenberger, R. L. and Stein, E. S. (1995) A simulation study of air traffic controllers' situation awareness. *Proceedings of an International Conference: Experimental Analysis and Measurement of Situation Awareness.* Daytona Beach, FL.

[Sorkin and Woods 1985] Sorkin, R. D. and Woods, D. D. (1985) Systems with human monitors: A signal detection analysis. *Human-Computer Interaction,* 1**,** 49-75.

[SourceForge 08] The SourceForge.net Open Source Software Development Web Site, available at http://sourceforge.net, accessed at 29 March, 2008

[Sousa, P., A. N. Bessani, et al. 2007]. "Resilient Intrusion Tolerance through Proactive and Reactive Recovery". Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), IEEE Computer Society.

[Spainhower, L. 2005]. An Overview of IBM Mainframe Dependable Computing : from System/360 to Series. Dependable Computing Systems : Paradigms, Performance Issues, and Applications. H. B. Diab and A. Y. Zomaya, Wiley-Interscience.

[Spainhower, L. and T. A. Gregg 1999]. "IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective." IBM Journal on Research and Development 43(5/6): 863-874.

[SPEC] Standard Performance Evaluation Corporation, http://www.spec.org.

[SPECWeb 99] Standard Performance Evaluation Corporation, "SPECweb99 Release 1.02 (Design Document)", http://www.spec.org/web99/, July 2000.

[Spitzner 2003] L. Spitzner: *Honeypots – Tracking Hackers* Addison-Wesley, 2003, ISBN 0-321-10895-7

[SRI 2008]. "Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD)." accessed Dec. 2008, from http://www.sdl.sri.com/projects/emerald/.

[Sridharan et al. 98] M. Sridharan, S. Ramasubramanian, and A. K. Somani. Himap: Architecture, features, and hierarchical model specification techniques. In Lecture Notes in Computer Science, number 1469, pages 348-351. R. Puigjaner, N. N. Savino, and B. Serra, 1998.

[Stancev, B. and L. Gavrilovska 2001]. "Implementation of Accounting Model within SNMPv3 Architecture". Proceedings of the Ninth IEEE International Conference on Networks, ICON 2001.

[StanfordLinearAccelerationCentre 2008]" Network Monitoring Tools" http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html

[Stanford-Clark, A. 2002]. Integrating Monitoring and Telemetry Devices as Part of Enterprise Information Resources, WebSphere MQ Development, IBM Software Group**:** 14.

[Staniford *et al.* 2002] S. Staniford, V. Pawson and N. Weaver, "How to Own the Internet in your Spare Time", in *USENIX Security Symposium,* pp.149-167, 2002.

[Stock 07] A. Stock, J. Williams, D. Wichers, "OWASP top 10", OWASP Foundation, July, 2007

[Stott 00] D. T. Stott, B. Floering, D. Burke, Z. Kalbarczyk, and R. K. Iyer, "Nftape: A Framework for Assessing Dependability in Distributed Systems with Lightweight Fault Injectors," in IEEE International Computer Performance and Dependability Symposium (IPDS'00), 2000, pp. 91-100.

[Strater et al 2001] Strater, L. D., Endsley, M. R., Pleban, R. J. & Matthews, M. D. (2001) Measures of platoon leader situation awareness in virtual decision making exercises (No. Research Report 1770). Alexandria, VA, Army Research Institute.

[Strater 2004] Strater, O. (2004) Human reliability analysis: data issues and errors of commission. *Reliability Engineering & System Safety,* 83**,** 127.

[Strunk, J. D., G. R. Goodson, et al. 2002]. "Intrusion Detection, Diagnosis, and Recovery with Self-Securing Storage". Pittsburgh, School of Computer Science, Carnegie Mellon University**:** 30.

[Swain 1987] Swain, A. D. (1987) Accident Sequence Evaluation Program Human Reliability Analysis Procedure. NUREG/CR-4772/SAND86-1996. Washington, DC, Sandia National Laboratories for the U.S. Nuclear Regulatory Commission.

[Swain 1990] Swain, A. D. (1990) Human Reliability-Analysis - Need, Status, Trends and Limitations. *Reliability Engineering & System Safety,* 29**,** 301-313.

[Swain and Guttman 1983] Swain, A. D. and Guttman, H. E. (1983) Handbook of human reliability analysis with emphasis on nuclear power plant applications. NUREG/CR-1278. Washington D.C.

[Swanson et al. 2003] Swanson M, N Bartol, J Sabato, J Hash, and L Graffo. July 2003. Security Metrics Guide for InformationTechnology Systems. NIST Special Publication 800-55, National Institute of Standards and Technology, Gaithersburg, Maryland. Available at http://csrc.nist.gov/publications/nistpubs/800-55/sp800-55.pdf.

[Takanen et al. 2008] Ari Takanen, Jared DeMott, Charlie Miller, "Fuzzing for Software Security Testing and Quality Assurance: Robustness Testing for Quality Assurance and Vulnerability", Artech House, ISBN 9781596932142, June, 2008.

[Taylor 1989] Taylor, R. M. (1989) Situational awareness rating technique (SART): The development of a tool for aircrew systems design. *Proceedings of the AGARD AMP Symposium on Situational Awareness in Aerospace Operations, CP478.* Seuilly-sur Seine: NATO AGARD.

[TCSEC 1985] TCSEC, Trusted Computer System Evaluation Criteria, Department of Defense, USA, 1985.

[Thomas 2005] Nigel Thomas. Performability of a Secure Electronic Voting Algorithm. Electronic Notes in Theoretical Computer Science 128(4), 22 April 2005, pp. 45-58, Proc. of the First International Workshop on Practical Applications of Stochastic Modelling (PASM 2004)

[Thonnard 2008] O. Thonnard, M. Dacier, “A framework for attack patterns' discovery in honeynet data”. DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA

[TimberlineTech 2008]. "Alphabetical List of Intrusion Detection Products." accessed Dec. 2008, from http://www.timberlinetechnologies.com/products/intrusiondtct.html.

[TimeRover 2009] "StateRover Programming Environment" accessed June 2009, from http://www.time-rover.com/staterover.pdf

[Tipton, H. F. and M. Krause 2003] "Information Security Management Handbook", CRC Press.

[Tixeuil 06] S. Tixeuil, W. Hoarau and L. Silva “An Overview of Existing Tools for Fault-Injection and Dependability Benchmarking in Grids” Technical Report TR-0041, The CoreGRID Network of Excellence, 2006, http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0041.pdf

[Toulmin 58] S E Toulmin “The Uses of Argument” Cambridge University Press, 1958 [TPC] Transaction Processing Performance Council, www.tpc.org.

[TPC-App 05] Transaction Processing Performance Council, “TPC Benchmark App (Application Server), Standard Specification, Version 1.5.0”, 2005, available at: http://www.tpc.org/tpc_app/.

[TPC-A 94] Transaction Processing Performance Council, “TPC Benchmark A, Standard Specification, Version 2.0”, 1994, available at: http://www.tpc.org/tpcc/.

[TPC-B 94] Transaction Processing Performance Council, “TPC Benchmark B, Standard Specification, Version 2.0”, 1994, available at: http://www.tpc.org/tpcc/.

[TPC-C 07] Transaction Processing Performance Council, “TPC Benchmark C, Standard Specification, Version 5.9”, 2007, available at: http://www.tpc.org/tpcc/.

[TPC-E 08] Transaction Processing Performance Council, “TPC Benchmark E, Standard Specification, Version 1.5.0”, 2008, available at: http://www.tpc.org/tpce/.

[TPC-H 08] Transaction Processing Performance Council, “TPC Benchmark H (Decision Support), Standard Specification, Version 2.6.2”, 2008, available at: http://www.tpc.org/tpch/.

[Trivedi 01] K. S. Trivedi, “Probability and Statistics with Reliability, Queuing, and Computer Science Applications”, John Wiley and Sons, New York, 2001.

[Trivedi 02] K. S. Trivedi. Sharpe 2002: symbolic hierarchical automated reliability and performance evaluator. In IEEE Int. Conference on Dependable Systems and Networks (DSN 2002), page 544, Washington, DC, 2002.

[Trivedi 96] K. S. Trivedi, S. Hunter, S. Garg and R. Fricks, “Reliability Analysis Techniques Explored Through a Communication Network Example”, Technical Report TR-96/32, Duke University, Dep. of Electrical and Computer Eng., USA, 1996.

[Tsai 05] W. T. Tsai , X. Wei, Y. Chen, R. Paul, “A Robust Testing Framework for Verifying Web Services by Completeness and Consistency Analysis”, Proceedings of the IEEE International Workshop on Service-Oriented System Engineering, October, 20-21, pp:159 – 166, 2005

[Tsai 96] T. K. Tsai, R. K. Iyer, and D. Jewitt, "An Approach Towards Benchmarking of Fault-Tolerant Commercial Systems," in 26th Int. Symp. on Fault Tolerant Computing, 1996, pp. 314-323.

[Tunstall 02] G. Tunstall, W. Clegg, D. Jenkins, C. Chilumbu, “Head-Media Interface Instability Under Hostile Operating Conditions,” IEEE Trans. on Instrumentation and Measurement, vol. 51, no. 2, April 2002 pp.293-298.

[Tyler et al 1998] Tyler, S. W., Neukom, C., Logan, M. & Shively, J. (1998) The MIDAS human performance model. *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting, Vols 1 and 2*, 320-324.

[UKHS 87] U.K. Health and Safety Executive. ``Programmable electronic systems in safety related applications'' 1987.

[UKHS 93] U.K. Health and Safety Executive, “Out of control” — a compilation of incidents involving control systems, (draft document), 1993.

[UKMoD 89] UK Ministry of Defence Draft Interim Def-Stan 00-55, The procurement of safety critical software in defence equipment, 1989.

[UKMoD 91] UK Ministry of Defence Interim Def-Stan 00-56. Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment, 1991

[UKMoD 04] UK Ministry of Defence. Def Stan 00-56, “Safety Management Requirements for Defence Systems”, Defence Standard 00-56/Issue 3, December 2004. [UPV] Universidad Politechnica de Valencia, Valencia – Spain, http://www.upv.es/.

[Urban et al 2001] P. Urbàn and X. Dèfago and A. Schiper. Neko: a Single Environment to Simulate and Prototype Distributed Algorithms. Proc. of the 15th Int'l Conf. on Information Networking (ICOIN-15).

[USNRC 2000] USNRC (2000) Technical basis and implementation guidelines for a technique for human event analysis (ATHEANA). NUREG-1624, Rev. 1. US Nuclear Regulatory Commission. Washington, D.C.

[Valeur et. al. 2004] F. Valeur, G. Vigna, C. Kruegel, and R.A. Kemmerer, “A Comprehensive Approach to Intrusion Detection Alert Correlation”, IEEE Trans. On Dependable and Secure Computing, Vol. 1, No. 3, 2004.

[Van der Veer 2008] van der Veer, G. C. (2008) Cognitive Ergonomics in Interface Design - Discussion of a Moving Science. *Journal of Universal Computer Science,* 14**,** 2614-2629.

[Van Moorsel, A. 2009] "State of the Art on Assessing, Measuring and Benchmarking Resilience", AMBER Coordination Action: 137.

[Van Moorsel & Huang 98] A. P. A. van Moorsel and Y. Huang. Reusable software components for performability tools, and their utilization for web-based configuration tools. In Proceedings of the 10th International Conference in Computer Performance Evaluation: Modeling Techniques and Tools, pages 37-50, Palma de Mallorca, Spain, September 1998. R. Puigjaner, N. N. Savino, and B. Serra.

[Van Renesse, R., K. P. Birman, et al. 2003]. "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining." ACM Transactions on Computer Systems (TOCS) 21(2): 164-206.

[Vandiver, B., H. Balakrishnan, et al. 2007] "Tolerating Byzantine Faults in Transaction Processing Systems Using Commit Barrier Scheduling". Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles, Stevenson, Washington, USA, ACM (NY, USA).

[Vargas 05] F. Vargas, D.L. Cavalcante, E. Gatti, D. Prestes, D. Lupi, "On the proposition of an EMI-based fault injection approach," 11th IEEE International On-Line Testing Symposium (IOLTS 2005), pp. 207-208, 2005

[Viera and Madeira 2002] Vieira, M. and Madeira, H. (2002) Recovery and Performance Balance of a COTS DBMS in the Presence of Operator Faults. *Proceedings of the 2002 International Conference on Dependable Systems and Networks.* IEEE Computer Society.

[Vieira 03a] M. Vieira and H. Madeira, "Benchmarking the Dependability of Different OLTP Systems", International Conference on Dependable Systems and Networks, DSN-DCC 2003, San Francisco, CA, USA, June 2003.

[Vieira 03b] M. Vieira and H. Madeira, “A Dependability Benchmark for OLTP Application Environments”, in Proceedings of the 29th International Conference on Very Large Data Bases, VLDB 2003, pp. 742-753, Berlin, Germany, 2003.

[Vieira 05] M. Vieira and H. Madeira, "Towards a security benchmark for Database Management Systems", Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, DSN2005, Yokohama, Japan, June 2005.

[Vieira 07] Vieira, M. and Laranjeiro, N. and Madeira, H. , "Benchmarking the Robustness of Web Services", The 13th IEEE Pacific Rim Dependable Computing Conference (PRDC 2007), Melbourne, Victoria, Australia, December 2007

[Vincenti 93] W. G. Vincenti, *What Engineers Know and How They Know it: Analytical Studies from Aeronautical History*: Johns Hopkins University Press, 1993.

[Vnunet 07] Vnunet, August, 2007, http://www.vnunet.com/vnunet/news/2197408/monster-kept-breach-secret-five

[Voas 98] J. M. Voas and G. McGraw, Software fault injection : inoculating programs against errors. New York: Wiley, 1998.

[Voas, J. 2000]. J. Voas, "Deriving Accurate Operational Profiles for Mass-Marketed Software". Proceedings of the 4th International Conference on Empirical Assessment & Evaluation in Software (EASE 2000). Keele University, Staffordshire, UK, Springer.

[Voas, J. 2008]. J. Voas, "A Schema for Collection of Operational Profile Data, US Patent 6862696." accessed October 2008, from http://www.patentstorm.us/patents/6862696/description.html.

[W3C 2008]. "Web Services Glossary." accessed Oct. 2008, from http://www.w3.org/TR/ws-gloss/.

[Wang 06] N. J. Wang and S. J. Patel, "Restore: Symptom-Based Soft Error Detection in Microprocessors," IEEE Transactions on Dependable and Secure Computing, vol. 3, pp. 188-201, 2006.

[Warm et al 1998] Warm, J. S., Dember, W. N. & Hancock, P. A. (1998) Workload and vigilance. *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting, Vols 1 and 2***,** 769-771.

[Watterson, C. and D. Heffernan 2007]. "Runtime Verification and Monitoring of Embedded Systems." IET Software 1(5): 172-179.

[Westrum 06] R. Westrum, "A Typology of Resilience Situations," in *Resilience Engineering. Concepts and Precepts*, E. Hollnagel, D. D. Woods, and N. Leveson, Eds. Aldershot, UK: Ashgate, 2006.

[Wickens 2002] Wickens, C. D. (2002) Situation awareness and workload in aviation. *Current Directions in Psychological Science,* 11**,** 128-133.

[Wickens and Kessel 1979] Wickens, C. D. and Kessel, C. (1979) Effects of Participatory Mode and Task Workload on the Detection of Dynamic System Failures. *IEEE Transactions on Systems Man and Cybernetics,* 9**,** 24-34.

[Wiener 1981] Wiener, E. L. (1981) Complacency: is the term useful for air safety? *26th Corporate Aviation Safety Seminar.* Denver, CO, Flight Safety Foundation, Inc.

[Williams 1986] Williams, J. C. (1986) HEART- a proposed method for assessing and reducing human error. *Ninth Advances in Reliability Technology Symposiumn.* NEC Birmingham, AEA Technology, Culcheth, Warrington.

[Williams, B. C., M. D. Ingham, et al. 2003]. "Model-based Programming of Intelligent Embedded Systems and Robotic Space Explorers." Proceedings of the IEEE 91(1): 212 - 237.

[Williams, B. C. and U. Nayak 1996]. A Model-based Approach to Reactive Self-configuring Systems. Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI-96, Portland, Oregon.

[Wilson 02] D. Wilson, B. Murphy and L. Spainhower. "Progress on defining Standardized Classes for Comparing the Dependability of Computer Systems" in Proceedings of the DSN 2002 Workshop on Dependability Benchmarking, pp. F1-5, Washington, D.C., USA, 2002.

[Wing 1998] J. Wang, "A Symbiotic Relationship Between Formal Methods and Security", School of Computer Science, CMU, 1998.

[Woods 06] D. D. Woods, "Essential Characteristics of Resilience," in *Resilience Engineering. Concepts and Precepts*, E. Hollnagel, D. D. Woods, and N. Leveson, Eds. Aldershot, UK: Ashgate, 2006, pp. 21-34.

[Woods and Wreathall 08] D. D. Woods and J. Wreathall, "Stress-Strain Plots as a Basis for Assessing System Resilience," in *Resilience Engineering Perspectives Volume 1: Remaining Sensitive to the Possibility of Failure*, *Resilience Engineering Perspectives*, E. Hollnagel, C. P. Nemeth, and S. Dekker, Eds.: Ashgate, 2008, pp. 145-161.

[Wreathall 1981] Wreathall, J. (1981) Human Reliability: Fact or Fiction? *Proceedings of the ANS/ENS Topical Meeting on PRA.* Lagrange Park, IL, American Nuclear Society.

[WSBang] WSBang, Trusted Partner in Security Testing, available at http://www.isecpartner.com/wsbang.html

[Xen 2008]. "The Xen® Hypervisor." accessed Dec. 2008, from http://www.xen.org/.

[Xu 99] J. Xu, Z. Kalbarczyk, and R. K. Iyer, "Networked Windows NT System Field Failure Analysis", in Proceedings of the IEEE Pacific Rim Dependable Computing Symposium, 1999.

[Young and Stanton 2004] Young, M. S. and Stanton, N. A. (2004) Taking the load off: investigations of how adaptive cruise control affects mental workload. *Ergonomics,* 47**,** 1014-1035.

[Yuste 03] P. Yuste, D. de Andres, L. Lemus, J. Serrano, and P. Gil, "Inerte: Integrated Nexus-Based Real-Time Fault Injection Tool for Embedded Systems," in Int. Conf. on Dependable Systems and Networks San Francisco, California, 2003, pp. 669-669.

[Zhao 2008] Yishi Zhao and Nigel Thomas. Approximate Solution of a PEPA Model of a Key Distribution Centre. Performance Evaluation: Metrics, Models and Benchmarks, LNCS 5119/2008, Springer, pp. 44-57, 2008.

[Zheng, Z. and M. R. Lyu 2008]. "WS-DREAM: A Distributed Reliability Assessment Mechanism for Web Services". Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, Anchorage, Alaska, USA.

[Zhu 03a] J. Zhu, J. Mauro and I. Pramanick. "R3 - A Framework for Availability Benchmarking," in Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2003, pp. B-86-87, San Francisco, CA, USA, 2003.

[Zhu 03b] Ji J. Zhu, J. Mauro, and I. Pramanick, "Robustness Benchmarking for Hardware Maintenance Events", in Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2003, pp. 115-122, San Francisco, CA, USA, 2003.

[Zou *et al.* 2005] C. C. Zou, W. Gong, D. Towsley and L. Gao, "The Monitoring and Early Detection of Internet Worms", *IEEE/ACM Transactions on Networking*, 13(5), pp.961-974, 2005.