Fault tolerance and network integrity measures: the case of computer-based systems

Peter Andras[*]
[*]School of Computing Science
University of Newcastle
peter.andras@ncl.ac.uk

Olusola Idowu[*]
[*]School of Computing Science
University of Newcastle
o.c.idowu@ncl.ac.uk

Panayiotis Periorellis[*]
[*]School of Computing Science
University of Newcastle
panayiotis.periorellis@ncl.ac.uk

Abstract

Fault tolerance is a key aspect of the dependability of complex computer-based systems. Fault tolerance may be difficult to measure directly in complex real world systems, and we propose here to measure it in terms of integrity preservation of the system under the assumption of a particular fault occurrence distribution. We measure the integrity preservation ability of the system by measuring the change of structural integrity of the graph representing the system while it is exposed to random node removal according to the assumed fault distribution. We show how to use such measures to measure the integrity reservation of computer-based systems and in this way indirectly their fault tolerance. We discuss the application of the proposed method in the context of a real world example, the Linux operating system. The results indicate that integrity preservation metrics can serve as an appropriate measure of fault tolerance of complex computer-based systems.

1 Introduction

The concept of fault tolerance of complex computer-based systems, and in particular of computers and software, emerged very early in the 1950s (Lee and Anderson, 1990). It was recognized that unexpected faults may emerge in computer-based systems, and that effective dealing with such faults it is critical for highly dependable systems. Fault tolerance is a key measure of the dependability of computer-based systems (Lee and Anderson, 1990; Laprie, 1992), dependability being defined as reliability, availability, safety, security, survivability, and maintainability of a system (Aviziensis et al, 2001).

Generally systems can be perceived as a set of units that are interconnected by their actions and behaviours (von Bertalanffy, 1973). Computer-based systems can be seen as systems with units which can be computer hardware, software, humans, and possibly a variety of other machines and human artefacts containing sensors and actuators. The interconnecting actions and behaviours of these units can take the form of data entry to the computers, data communications between hardware components, data interchange and processing by software components, and display or communication of data to actuators.

An interesting issue is how to measure the fault tolerance of a computer-based system. Systematic mathematical analysis of fault tolerance of models of computer-based systems started in 1960s (Lee and Anderson, 1990). Typically fault tolerance is evaluated by full probabilistic analysis of the system, by calculating measures such as mean time to failure and mean time to repair under the assumption of a fault occurrence scenario (e.g., identical and independent fault occurrence distribution for each system component) (Amari, 2000; Chang et al, 2004; Ou and Dugan, 2003; Scerrer and Steininger, 2003).

One stream of fault tolerance research is focused on the analysis of graphs that represent computer-based systems (Billinton and Jonnavithula, 1999; Bell, 2003; Cheng and He, 2004). These works assume a fault occurrence scenario in the graph (e.g., node failure or edge failure) and measure the probability of connectedness (Beichelt and Tittmane, 1991; Elperin et al, 1991) or of having flow capacity above a given limit (Chan et al, 1997; Kishimoto, 1997) of the graph as a proxy measure for the fault tolerance of the system represented by the graph. The main drawback of these methods is that they are very computationally intensive and in many cases they are restricted to a narrow range of particular graph topologies (Al-Sadi et al, 2002; Goerdt, 2001; Goerdt and Molloy, 2003).

An alternative way to analyse the robustness of systems is to use structural graph analysis methods that reveal vulnerable components and the sensitivity to structural damage of the system (Albert et al, 1999). These methods assess the integrity of the system and the change of integrity measures after structural damage to the system in terms of structural measures, such as diameter, average minimum path length or average clustering coefficient. The underlying theoretical assumption is that system structural integrity implies functional integrity of the system (Andrews and Beeson, 2003; Ferrandi et al, 2003). This is supported by practical examples, which show that structural integrity and functional integrity of systems are strongly correlated (Albert et al, 1999; Jeong et al, 2001). Consequently, the analysis of the structural integrity of the graph representing a system by appropriate structural measures can provide indicator measures of the functional integrity of the system.

We propose in this paper the use of structural graph analysis methods to measure the integrity of computer-based systems. We measure the likely structural damage as an approximation of likely functional damage due to the presence of faults. In this way we can assess the fault tolerance of the system by measuring the likely change of structural integrity of the system.

The rest of the paper is structured as follows. Section 2 discusses system integrity measures. In Section 3 we analyse the link between fault tolerance and integrity measures. Section 4 presents an example of the application of the proposed methodology to the assessment of fault tolerance of computer-based systems. Finally, in section 5 we draw some conclusions of the paper.

2 System integrity

Systems are sets of component units interconnected by their interactions (von Bertalanffy, 1973). Component units interact by their behaviour modifying the state of the units participating in such interactions. In a stronger sense we may consider systems as only those sets of interacting component units, in which the interactions between components depend primarily on earlier interactions between system components (Andras and Charlton, 2005). We should also point out that system components may also interact with other units, which are not part of the system. Such interactions constitute the system's interaction with its environment.

The integrity of a system can be defined in functional terms as the system's ability to perform the full range of system behaviours (Ferrandi et al, 2003). The system behaviours are possible patterns of behaviours of its component units (Lee and Anderson, 1990). Some of these behaviours may have an effect on the system's environment, while others may cause only a change of the internal behaviour of the system.

Measuring functional integrity directly may be difficult, as the full range of possible system behaviour may not be known (Ferrandi et al, 2003). A way to approximate the functional integrity of a system is to measure its structural integrity ((Andrews and Beeson, 2003). In practical cases of living cells (Jeong et al, 2001), nervous systems of animals (Scannell et al, 1995), and technological systems (Albert et al, 1999) it has been shown that their functional integrity correlates strongly with their structural integrity. Measuring structural integrity is much simpler than measuring functional integrity, in the sense that it requires only the measurement of the existence of components and interactions between components, disregarding the actual functional semantics of interactions and interaction patterns.

Structural integrity measures of systems are based on the measurement of the structural integrity of the underlying graph structure of the system, which is made of nodes representing system units, and edges or arcs representing undirected or directed interactions between system units. (We consider undirected graphs only in what follows).

Simple measures of structural integrity of graphs include the diameter, the average minimum path length and the average clustering coefficient of the graph. The diameter is defined as the largest of the minimal path lengths between nodes of the graph:

$$D(G) = \max\{l(i,j) \mid i,j \in V\}, G = \{V,E\} \quad (1)$$

where V is the set of nodes and E is the set of edges of the graph, and l(i,j) is the minimal length of a path between the nodes i and j. The average minimum path length is defined as the average length of minimal paths between all pairs of nodes of the graph that can be connected (i.e. infinite length shortest paths are ignored):

$$\mu = \frac{1}{|V|^2} \sum_{\substack{i,j\in V \\ l(i,j)<\infty}} l(i,j) \quad (2)$$

where we use the same notations as above. The clustering coefficient of a node is the proportion between the number of existing edges between the neighbours of the node and the number of all possible edges:

$$c(i) = \frac{2 \cdot |\{(i,j) \mid (i,j) \in E\}|}{|\{j \mid (i,j) \in E\}| \cdot (|\{j \mid (i,j) \in E\}| - 1)}, i \in V \tag{3}$$

The average clustering coefficient of the graph is the algebraic average of the clustering coefficients of all nodes:

$$\eta(G) = \frac{1}{|V|} \sum_{i \in V} c(i) \tag{4}$$

We note that the above measures evaluate somewhat different aspects of the graph integrity; none of them provides a comprehensive evaluation of the graph integrity. In order to be on the safe side in practical applications the best practice is to use such a set of simple integrity measures and evaluate the graph integrity using the resulting set of integrity measure values (i.e., by considering a vector of integrity measure values). In particular, if we need a single value measure of the graph integrity on the basis of a vector of integrity measures, the safest is to take the value indicating the greatest amount of integrity loss.

Other more sophisticated measures of graph integrity include the calculation of coefficients of the graph's characteristic polynomial, and eigenvalues of the graph's adjacency matrix. These methods can provide a full picture of the graph's integrity and in principle capture all its aspects. The disadvantage of these methods is that they are computationally very expensive, and the calculation of the required numbers may be impractical for very large graphs representing complex systems. The above introduced simple integrity measures are well correlated with the more general measures. The largest eigenvalue of the adjacency matrix is related to the density of the edges, the second eigenvalue is related to the conductivity within the network (Farkas et al, 2001). The second coefficient of the characteristic polynomial is related to the number of edges, while the third coefficient is twice the number of triangles in the network (Biggs, 1994).

An important issue regarding the use of graph integrity measures to assess the integrity of systems is that of how to actually measure the system components and their interactions. One approach can be to consider the design of the system, if this is available. (For technological systems this might often be the case.) However, this approach can lead one to fall into the trap of showing the robustness of the designed system and not of the actual system. We believe that the right approach is to measure the existing components of the real system and their existing interactions in order to assess the integrity and robustness of the actual system. However, we recognize that in some practical cases such measurements might prove to be difficult (e.g., monitoring of human – computer interactions), limiting the applicability of the structural graph analysis based assessment of system integrity evaluation.

In the case of computer-based systems we typically have a set of non-computer related units (e.g., humans, sensors), a set of hardware units making the computer hardware part of the system, and usually a very large set of software modules, constituting the units of the system. In some cases we ignore the non-computer related units and even the hardware part of the system and we focus our attention exclusively on the system made of software units. The interactions between software units take the form of data transactions between them, which can be measured by appropriate monitoring of the system (Periorellis et al, 2004).

3 Fault tolerance and integrity preservation

Faults are unexpected behaviours of system components. Faults in computer-based systems may have a number of origins; they can be classified as design faults, physical faults and interaction faults (Aviziensis et al, 2001). Faults cause errors in the system, which are deviations from the expected behaviour of the system. Errors in computer-based systems may stay latent, until they are detected, when they cause abnormal behaviour at the interface of the system with its environment (Lee and Anderson, 1990). Errors cause failures of the system, when the system is unable to perform its function correctly (Aviziensis et al, 2001).

Faults in the system may occur at various places. An important feature of faults is their occurrence within the system and their distribution at these places within the system. In many cases we may suppose that the faults may appear at any system unit according to the same occurrence distribution, no unit being more susceptible for producing faulty interactions than others (Amari, 2000). In some cases we may also use the hypothesis that the likelihood of faulty interactions is proportional with the likelihood of the unit being involved in interactions. In other cases the fault distribution may follow some peculiar well defined distribution, such as in the case of faults induced by malicious logic (e.g., attacks by hackers). The types of fault distributions determine the fault occurrence environment of the system.

Fault tolerance is the ability of the system to maintain its functionality in the presence of active faults (Lee and Anderson, 1990). Fault tolerance is typically achieved by error detection, recovery and fault handling (Aviziensis et al, 2001). Fault tolerance of computer-based systems depends on the fault occurrence environment of the system (e.g., in presence of naturally occurring faults the system may prove sufficiently fault tolerant, while in the presence of targeted attack by hackers, it may prove fault sensitive).

In general the measure of fault tolerance of the system is a relative measure, which shows to what extent the system preserves its functionality in a certain fault occurrence environment (Lee and Anderson, 1990). To assess the fault tolerant nature of a computer-based system we need to assess the level of functionality of the system within the considered fault occurrence environment. In other words we need to evaluate the functionality preservation ability of the system. Usually some probabilistic approach is used to evaluate fault tolerance measures such as mean time to failure or mean time to repair (Lee and Anderson, 1990). These methods take into consideration the whole system resulting computationally very intensive analyses in case of large systems (Chang et al, 2004; Billington and Jonnavithula, 1999). To perform such exhaustive evaluations may prove difficult in practice, as monitoring and assessing all aspects of the functionality of the system and performing all the required calculations may be extremely time and resource consuming (Ferrandi et al, 2003). Alternative methods were proposed recently, involving game theoretic approaches (Bell, 2003), formal languages inspired analysis (Phoha et al, 2004) and structural network analysis approaches (Albert et al, 1999).

We adopt the structural network analysis approach, which has the key advantage that it implies a relatively low computational load for the evaluation of large systems. We measure the fault tolerance of the system by evaluating the ability of the system to preserve its integrity. The measure of integrity preservation is calculated by using system integrity measures based on a structural graph analysis of the graph representing the system. As structural integrity is strongly correlated with functional integrity, the structural integrity preservation measure provides a proxy measure of the functional integrity preservation measure of the system. Consequently, we can use the structural integrity measures introduced in the previous section to measure the change of the integrity of the system in a given fault occurrence environment.

To measure the effects of faults on the integrity of the system, we simulate the faults by sampling the fault occurrence distributions and then evaluating the integrity measures of the system in the presence of simulated faults. The presence of faults causes the elimination from the graph of the system of edges between nodes or of nodes of the graph. These changes happen according to the fault occurrence distributions and have the effect that the integrity measures of the system graph are modified. The expected changes in terms of integrity measures may be calculated analytically in the case of small systems or can be evaluated by numerical simulations in the case of large and complex systems. The expected changes associated with a fault occurrence environment characterise the system's integrity preservation ability and are used as an approximate measure of the fault tolerance of the system.

To show how to use the calculated integrity measures to assess the fault tolerant nature of a system we consider below a toy example. Let us consider a software system of 1000 units of which corresponding graph representation is shown in Figure 1. The system's structural integrity measures are the following: (1) diameter: D(S)=19; (2) average minimum path length: $\mu(S)=3.58151$; (3) average clustering coefficient: $\eta(S)=0.022702$

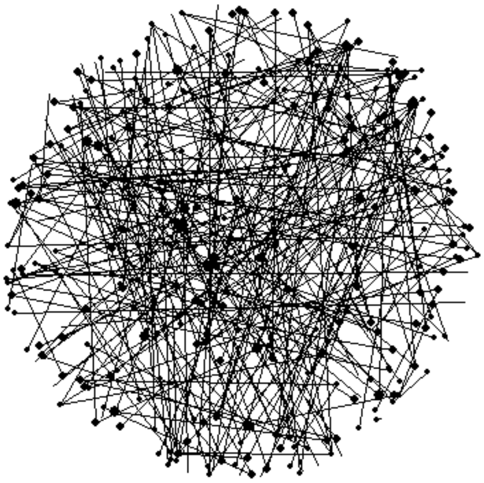

Figure 1: The graph representation of the model system with 1000 nodes. The size of the nodes indicates the number of connections of the node. Only the subset of more connected nodes and the subset of connections between these nodes are displayed to keep the figure comprehensible.

We consider a fault occurrence environment in which the faults occur with equal uniform probability (p=0.15) at each unit of the system and each fault temporarily knocks out the system unit where it occurs. To evaluate the fault tolerance of the system we perform a numerical simulation of the fault occurrences, and evaluate the integrity measures of the system for each simulation. After the simulations we calculate the average values and variances of the system integrity measures. We chose to run 20 simulations in order to get reliable estimates of mean values (the variance of the mean value calculated from n measurements is σ/square-root(n), where σ is the variance of the calculated values). The calculations after the simulations led to the values: (1) diameter: avg(D(S))=23.05, var(D(S))=4.1355; (2) average minimum path length: avg(μ(S))=3.6994, var(μ(S))=0.0397; (3) average clustering coefficient: avg(η(S))=0.023, var(η(S))=0.0014.

To evaluate the integrity preservation ability of the system we calculate first, whether the average values of system integrity measures after the simulation of faults differ significantly or not from the corresponding values calculated for the fully functional system. Next we calculate the normalized distance of the pre-damage and post-damage integrity measure values, which together with their attached statistical significance levels characterize the fault tolerance of the system. In order to be on the safe side, we choose the worst measure (i.e., the largest and most significant distance) to be the numerical evaluation of the fault tolerance of the system. In the case of the above system the normalized distances (z-score, i.e., the distance measured between the mean value and original value in units equal to the standard deviation – $(v_{original} - m)/\sigma_m$) and statistical significance levels (statistical significance levels show how likely is that the original value is the same as the estimated mean value after damage, low p-value indicates that the likelihood of them being the same is very low, or in other words the two values differ significantly) are listed in Table 1.

In the case of the above toy example we have shown how to apply in principle the proposed structural graph analysis based integrity evaluation methods to assess the fault tolerance of a computer-based system. The data shown in the last column of Table 1 shows the values of the likelihoods that original value of the integrity measure is the same as its value after the damage. The results indicate that under the above described fault occurrence environment assumption the system suffers significant damage (p<0.01) in terms of diameter and average shortest path length, the amount of the latter damage being more significant than the former. Considering the most significant damage (i.e., the damage in terms of average path length, p=7.29 x 10^{-14}), we conclude that under the considered fault occurrence assumption the system represented by the graph suffers very significant structural and functional damage, and consequently has low fault tolerance.

Table 1: Summary of integrity measures of the system before and after damage, including the z-score for the original values considering the mean and variance of the after damage values (z-score = (original – damage mean)/(damage variance / square-root(20))), and the statistical significance level of the difference between the original values and the mean values calculated after the damage. The p values above 0.1 are omitted.

Integrity measure	Original	Damage mean	Damage variance	z-score	p-value
Diameter	19	23.05	4.135	4.379	1.2 x 10^{-5}
AvShortPath	3.582	3.699	0.040	13.26	7.29 x 10^{-14}
AvClusCoef	0.023	0.023	0.002	1.079	-

4 Application

Linux is one of the most popular operating systems, which is due to a good extent to its open source based development. It is commonly claimed that Linux is more reliable and secure than many other operating systems. An immediate question is how fault tolerant is Linux actually.

We analysed the network structure of the Linux under typical running conditions with a set of usual programs running. To perform the analysis we considered the calls between the classes present in the Linux kernel (version 2.4.19). We found 6815 classes and 19909 calls between them, by parsing the source code of the classes. The interaction network of the classes (see Figure 2) was then analysed in terms of structural network analysis.

Analysing the connectivity distribution of the processes we found that the distribution follows a power law distribution (with exponent $\gamma = -1.33$; see Figure 3) similar to the case of the Internet (Albert et al, 1999). This indicates that among the Linux classes there are relatively few very highly connected classes (which call and are called by many other classes) and many others with relatively few connections. This implies that similarly to the Internet (Albert et al, 1999) the Linux is very robust and fault tolerant if faults happen randomly following a uniform fault distribution over the processes (run-time representation of classes), while it should be very vulnerable and fault sensitive if faults are distributed such that they affect mostly the most highly-connected processes.

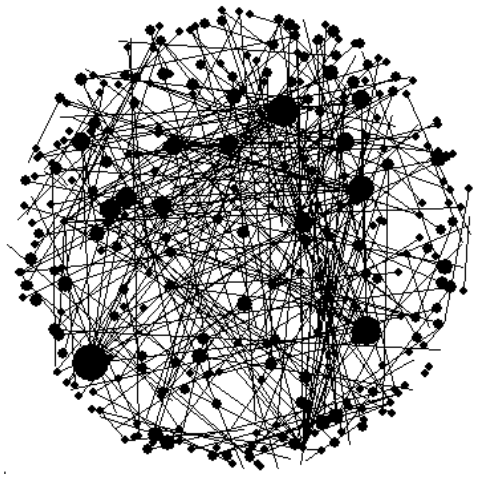

Figure 2: The graph representation of the Linux. The size of the nodes indicates the number of connections of the node. Only the subset of more connected nodes and the subset of connections between these nodes are displayed to keep the figure comprehensible.

We performed an analysis of the Linux class network to evaluate the effects of faults on integrity measures. We simulated a scenario with uniform random distribution with probability of faults at each node. We also performed a simulated a scenario when the likelihood of a node being faulty was proportional with the connectivity of the node. The analysis results are shown in Table 2 and Table 3.

The results show that as we expected Linux is remarkably fault tolerant in a fault occurrence environment characterised by uniform fault distribution, while it is significantly more fault sensitive in the case of a fault distribution centred on the mostly linked processes. This suggests that indeed the common belief about the reliability and fault tolerance of Linux is well founded in case of random uniformly distributed errors, but also highlights that Linux is also a vulnerable system in case of well designed malicious attacks. (Note that the tables show that the average shortest path is decreasing after damage. This is because infinite shortest paths between nodes belonging to isolated sub-networks are ignored.)

Table 2: Summary of integrity measures of Linux before and after random damage, including the z-score for the original values considering the mean and variance of the after damage values (z-score = (original – damage mean)/(damage variance / square-root(20))), and the statistical significance level of the difference between the original values and the mean values calculated after the damage. The p values above 0.1 are omitted.

Integrity measure	Original	Damage mean	Damage variance	z-score	p-value
Diameter	44	38.85	4.869	4.729	2.25 x 10^{-6}
AvShortPath	12.01	11.50	0.740	3.077	0.0021
AvClusCoef	0.133	0.133	0.007	0.350	-

Table 3: Summary of integrity measures of Linux before and after targeted damage, including the z-score for the original values considering the mean and variance of the after damage values (z-score = (original – damage mean)/(damage variance / square-root(20))), and the statistical significance level of the difference between the original values and the mean values calculated after the damage. The p values above 0.1 are omitted.

Integrity measure	Original	Damage mean	Damage variance	z-score	p-value
Diameter	44	35.45	5.062	7.552	1.41 x 10^{-13}

AvShortPath	12.01	10.47	1.444	4.766	1.87 x 10^{-6}
AvClusCoef	0.133	0.132	0.005	1.018	-

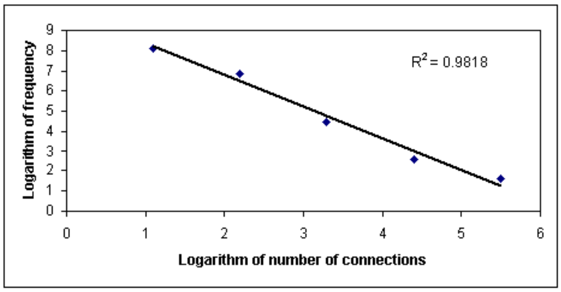

Figure 3: The distribution of connectivities in the case of the Linux classes and calls network

4 Conclusions

Fault tolerance is measurable aspect of the dependability of computer-based systems. Direct measurement of fault tolerance of large real world systems poses considerable problems, considering that most existing work is focused on exhaustive analytical evaluation of relatively simple model systems (Billinton and Jonnavithula, 1999; Bell, 2003; Cheng and He, 2004). An approach to find a proxy measure for the fault tolerance of large systems is to measure their structural integrity preservation under the assumption of a fault occurrence environment. This measure is based on the assumption that functional integrity is strongly correlated with structural integrity (Andrews and Beeson, 2003; Ferrandi et al, 2003), which is supported by experimental analysis of various real world complex systems (Albert et al, 1999; Jeong et al, 2001).

Integrity and integrity preservation of large systems can be measured by structural graph analysis of their graph representation, where nodes represent system units and edges represent interactions between system units. Simpler (e.g., average minimum path length) or more complex (e.g., eigenvalues of the adjacency matrix) measures can be used to estimate the structural integrity preservation of the system while exposed to faults occurring in accordance with an assumed fault distribution. Using these measures we can evaluate the likely amount of loss of integrity (or integrity damage) and the statistical significance of this loss.

A toy example and a real world example were presented to show the application of this fault tolerance measurement approach. These examples show that indeed the proposed methods can be applied effectively and lead to meaningful conclusions about the analysed systems. In the case of the real world example (Linux) the analysis indicates that the system is very fault tolerant under the assumption of uniform fault distribution, but that, not surprisingly, it is very vulnerable under the assumption of a fault distribution driven by properly targeted malicious interventions. However, the methods discussed here enable the extent of this effect to be analyzed in useful detail.

We believe that using relatively simple network integrity measures can simplify considerably the effective analysis of fault tolerance of large real world computer-based systems. Although these methods do not provide an exact measure of fault tolerance they provide good approximations of the actual measure. Such approximate measures can be used to rapidly determine the effects of a variety of fault occurrence environments, allowing the designers and developers of large systems to prepare appropriate defence and repair strategies to support the dependability of their system effectively and efficiently.

Acknowledgement

The authors thank Brian Randell for very helpful comments.

References

A. Aviziensis, J.C. Laprie, B. Randell. Fundamental concepts of dependability. *Newcastle University Report no. CS-TR-739* 2001.

R. Albert, H. Jeong, A.-L. Barabási. Diameter of the World Wide Web. *Nature*, 401:130-131, 1991.

J. Al-Sadi, K. Day, M. Ould-Khaoua. Unsafety vectors: a new fault-tolerant routing for the binary n-cube. *Journal of Systems Architecture*, 47:783-793, 2002.

S.V. Amari. Generic rules to evaluate system-failure frequency. *IEEE Transactions on Reliability*, 49:85-87, 2000.

REPLACE Charlton, B, Andras, P: The nature and function of management - a perspective from systems theory. Philosophy of Management 2004; 3: 3-16.

J.D. Andrews and S. Beeson. Birnbaum's measure of component importance for noncoherent systems. *IEEE Transactions on Reliability*, 52:213-219, 2003.

F. Beichelt and P. Tittmann. A generalized reduction method for the connectedness probability of stochastic networks. *IEEE Transactions on Reliability*, 40:198-204, 1991.

M.G.H. Bell. The use of game theory to measure the vulnerability of stochastic networks. *IEEE Transactions on Reliability*, 52:63-68, 2003.

N. Biggs. *Algebraic Graph Theory*. Cambridge, Cambridge University Press, 1994.

R. Billinton and S. Jonnavithula. Calculation of duration, frequency, and availability indexes in complex networks. *IEEE Transactions on Reliability*, 48:25-30, 1999.

Y. Chan, E. Yim, A. Marsh. Exact and approximate improvement to the throughput of a stochastic network. *IEEE Transactions on Reliability*, 46:473-486, 1997.

Y.-R. Chang, S.V. Amari, S.-Y. Kuo. Computing system failure frequencies and reliability importance measures using OBDD. *IEEE Transactions on Computers*, 53:54-68, 2004.

Y. Cheng and Z. He. Bounds on the reliability of distributed systems with unreliable nodes & links. *IEEE Transactions on Reliability*, 53:205-215, 2004.

T. Elperin, I. Gertsbakh, M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40:572-581, 1991.

I.J. Farkas, I. Derenyi, A.-L. Barabasi, T. Vicsek. Spectra of Real-World Graphs: Beyond the Semi-Circle Law. *arXiv*:cond-mat/0102335, 2001.

F. Ferrandi, F. Fummi, G. Pravadelli, D. Sciutto. Identification of design errors through functional testing. *IEEE Transactions on Reliability*, 52:400-412, 2003.

A. Goerdt. Random regular graphs with edge faults: expansion through cores. *Theoretical Computer Science*, 264:91-125, 2001.

A. Goerdt and M. Molloy. Analysis of edge deletion processes on faulty random regular graphs. *Theoretical Computer Science*, 297:241-260, 2003.

H. Jeong, S.P. Mason, A.-L. Barabasi, Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41-42, 2001.

W. Kishimoto. Reliable flow with failures in a network . *IEEE Transactions on Reliability*, 46:308-315, 1997.

J.C. Laprie (ed). *Dependability: basic concepts and terminology in English, French, German, Italian and Japanese*. Wien, Springer-Verlag, 1992.

P.A. Lee and T. Anderson. *Fault Tolerance. Principles and Practice*. Wien: Springer-Verlag, 2nd ed, 1990.

Y. Ou and J.B. Dugan. Approximate sensitivity analysis for acyclic Markov reliability models. *IEEE Transactions on Reliability*, 52:220-230, 2003.

P. Periorellis, O.C. Idowu, S.J. Lynden, M.P. Young, P. Andras. Dealing with complex networks of protein interactions: A security measure. In *Proceedings of 9th IEEE International Conference on Engineering of Complex Systems (ICECCS)*, P. Bellini, S.A. Bohner, B. Steffen (eds) pp 29-36, IEEE Computer Society, 2004.

V.V. Phoha, A.U. Nadgar, A. Ray, S. Phoha. Supervisory control of software systems. *IEEE Transactions on Computers*, 53:1187-1199, 2004.

J.W. Scannell, C. Blackmore, M.P. Young. Analysis of connectivity in the cat cerebral-cortex. *Journal of Neuroscience*, 15:1463-1483, 1995.

C. Scherrer and A. Steininger. Dealing with dormant faults in an embedded fault-tolerant computer system. *IEEE Transactions on Reliability*, 52:512-522, 2003.

L. von Bertalanffy. *General System Theory: foundations, development, applications*. Harmondsworth, Penguin, 1973.