

A Bayesian Network Approach to Modeling IT Service Availability using System Logs

Rui Zhang*

Eric Cope[†]

Lucas Heusler[‡]

Feng Cheng[§]

Abstract

The complexity of today's IT systems makes capturing the behaviors of these environments increasingly difficult, and calls for automated modeling solutions. We present an approach to generating a probabilistic (Bayesian) network for modeling IT service availability on information reported in system logfiles, including service desk problem tickets, configuration management databases and system event monitoring logs. In particular, we harvest these data to derive a Bayesian network structure that captures failure causality between system components and subsequently use the problem tickets to quantify the network parameters. Experiments based on a major European bank deployment have shown that our approach is able to generate models of reasonable accuracy even in the presence of limited amounts of data.

1 Introduction

Enterprise IT systems continue to grow rapidly both in size and in complexity. The increase in size inevitably leads to a higher number of self-inflicting failures on individual components (e.g. a bug in an application component or a physical server crash due to overheating), whereas a rise in complexity means more dense and unobvious interactions among software and hardware components, which may lead to a cascading series of failures. Understanding the system-wide implications of these failures is critical, as many of these IT systems are committed to delivering three nines (99.9%) or even five nines (99.999%) availability [14]. From a system management stand point, it allows for (i) deriving distributions with respect to the frequency

and duration of application/service outages. (ii) what-if analysis (in terms of outages) for system re-configuration, either hardware upgrade or topology change due to consolidation efforts and (iii) assessing the vulnerability of individual IT resources. In this paper, we shall be concerned with using data from systems logs and other sources to create a Reliability Bayesian Network (RBN) for the purpose of quantifying system and service availability, performing root cause analysis, and scenario assessment such as component failures and system reconfigurations.

Manually building analytical models to capture complicated failure dynamics is becoming less attractive, as it can be expensive, inaccurate and error-prone. With the advent of systems monitoring software and outage/failure logging infrastructure in today's IT deployments, it may become possible to extract sophisticated outage patterns from these data in an automated fashion. Herein we leverage configuration and event logs to discover the causal linkages from low-level component failures to high-level application/service disruptions via a readily available understanding of how IT failures spread through an environment; and then use problem tickets (service outage logs) to estimate the probabilities that failures will spread across each of these linkages.

The remainder of this paper is organized as follows. The sequel introduces an example IT environment referenced throughout the paper. The availability model is then detailed in Section 3. Experimental results with regard to the model's performance are reported in Section 4. The last two sections discuss related literature and future work.

2 Douglas: An Example IT environment

Douglas is the IT infrastructure and services of a major European bank, supporting Internet banking and brokerage transactions. It is a mid-size data center deployment consisting of over 100 virtual machine (VM) images, 20 strong physical servers, spanning 2 locations and implementing a collection of logically dependent IT services. A fraction of the Douglas system topology is displayed in Figure 1, which we shall use as a running example throughout this paper.

Figure 1 includes IT services, *Logon and Securities, Or-*

*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA, rui Zhang@us.ibm.com

[†]IBM Zurich Research Lab, Säumerstrasse 4, 8803 Rüschlikon, Switzerland, erc@zurich.ibm.com

[‡]IBM Switzerland, Vulkanstrasse 106, CH-8010 Zurich, Switzerland, lucas.heusler@ch.ibm.com

[§]IBM T.J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598, USA, fcheng@us.ibm.com

der Book as well as Buy and Sell . Logically, the Buy an Sell service is dependent on Order Book, which in turn relies on Logon and Securities to function. Each service is implemented in a distributed fashion by one or more application components, that is, user requests are directed to one of these application components according to their banking data needs. The application components utilize middleware (DBs) and run on VM images. The VM images are contained within physical machines connected by two different network segments and belonging to two sites.

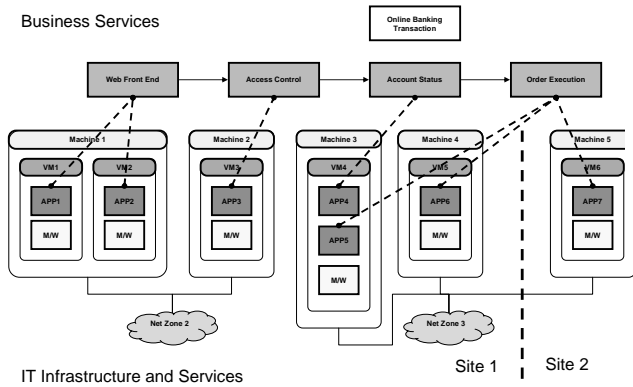


Figure 1. An excerpt of Douglas

The failure causality within such a small set of dependencies can already be complex. For example, consider the following potential service outage scenario regarding the Buy and Sell service. A serious bug within the APP5 application on image VM4 will impact the Buy and Sell service, yet at first glance, a complete unavailability of the service appears unlikely as many of its transactions are routed to the other two application components. On the other hand, depending on its nature the bug might corrupt the entire image VM4, which may bring down the only application component supporting the Order Book service and result in the loss of this entire service. Note that the Buy and Sell service can not logically function without Order Book and will thus become unavailable after all.

Figure 2 illustrates some service outage incidents recorded for Douglas over a 6 month period. Every row is a service outage incident reported to the help desk. It consists of an incident number (PMOG#), the incident duration (which is in fact used as the base of our mean-time-to-recovery (MTTR) calculations later in this paper), and the root cause isolated for this incident. More importantly, the record contains a system-wide snapshot of the up or down states of all components in the environment.

PMOG#	Duration (h)	RootCauseName	APP1_VM1	APP2_VM2	APP3_VM3	APP4_VM4
44033	1.665331	APP6_VM5	0	0	0	0
44075	0.004551	APP5_VM4	0	0	0	1
44177	1.205014	MACHINE2	0	1	0	0
44178	0.511925	APP6_VM5	0	0	0	0
44297	0.487341	0	0	0	0	0
44316	0.565098	MACHINE2	0	1	0	0
44317	0.684915	MACHINE4	0	0	0	0
44407	1.05199	MACHINE4	0	0	0	0
44463	0.153571	MACHINE2	0	1	0	0
44672	0.262141	MACHINE2	0	1	0	0

Figure 2. Outage record data.

This paper aims to estimate a probabilistic causal model for an environment like Douglas, using its configuration information (Figure 1) as well as outage records (Figure 2).

3 Availability Modeling

The key to capturing availability is a model that describes the reliability characteristics of the environment (i.e. up/down probability of a service or system component). In this section, we detail how Bayesian network theory can be used to model the system and learn its structure and parameters from service outage logs. Bayesian networks are adopted for this purpose because they (i) provide a probabilistic framework for representing complex, nondeterministic systems behaviors, (ii) can naturally encode knock-down causality among IT components and services, and (iii) can be easily understood by domain experts.

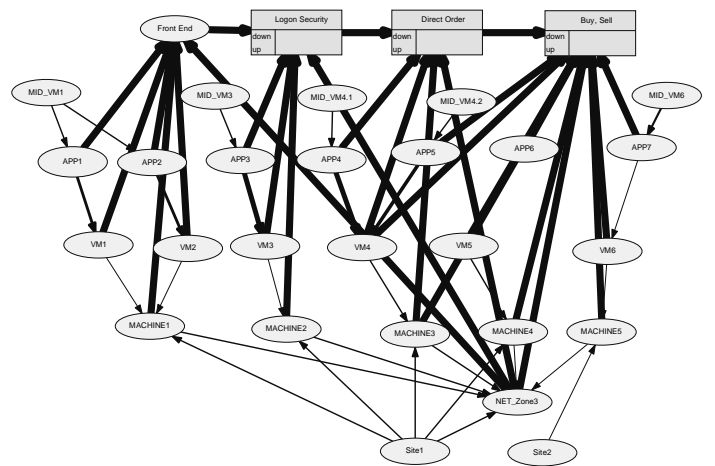


Figure 3. A RBN structure for Douglas. Thick arrows indicate strong failure causality.

A Reliability Bayesian Network (RBN) is a directed acyclic graph (DAG), consisting of a set of binary nodes

A_1, \dots, A_n each representing an IT element (component or service) being “up” (0) or “down” (1) and a set of direct arcs $\{< A_i, A_j > | i \neq j\}$ which in this model represent a stochastic causal relationship flowing from the parent to the child node, i.e., failure of a parent node could cause the child node to fail. The Bayesian network modeling paradigm is extremely flexible, allowing incorporation of a wide array of variables and relationship types. We present here a simple example model for quantifying service availability that has been shown to be realistic and tractable. An example RBN structure for the Douglas infrastructure is depicted in Figure 3. Section 3.1 discusses how the network structure can be derived from system information sources.

Each node A_i in the RBN is associated with an inherent failure probability IR_i and a set of transfer probabilities $\{p_{i1}, \dots, p_{im}\}$ with p_{ik} encoding how likely it is that component E_i will fail, given that its k -th parent has failed. The inherent failure probability is interpreted as the probability that a component will fail on its own within a fixed time period, such as a day. The availability of an element is determined jointly by inherent failures of that element and knock-down failures from elements with a direct causal relation. In this case, the probability that an element E is available equals

$$Pr\{E = 0\} = Pr\{IR = 0\} \times \prod_j ((1 - p_j)Pr\{PN_j = 1\} + Pr\{PN_j = 0\})$$

This relationship is illustrated in Figure 4. In Bayesian network terminology, each node is a “noisy-OR” node, which effectively assumes an independence of causal influence among the parent nodes on the child node. This assumption can be justified if there is no evidence regarding interactions on the influence on a child node if parent nodes fail simultaneously, or if parents rarely fail simultaneously. More complex interactions can also be encoded in the Bayes net framework, although it may come at the expense of model tractability.

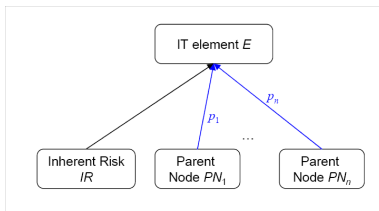


Figure 4. Noisy-OR causal dependence.

Subsection 3.1 explains how the RBN structure, transfer and inherent probabilities may be acquired from log data.

Once that is completed, we may use standard inferential techniques for Bayesian networks [7] to compute the probability f_i that any service and/or IT component i will fail within a given time period. Furthermore, given that any particular component or service is unavailable, we can calculate from the network the probability p_j that any particular root cause j may be to blame. If a root cause node is used, then we just determine this from the posterior probability of this node being in any one of its states. To determine the expected duration of a component failure given that a failure occurs (MTTR), we use the following formula:

$$D_i = \sum_{j \in E} p_{ij} \cdot d_j,$$

where E is the set of all risk events, p_{ij} is the probability that the root cause node takes the value j given that component i has failed, and d_j is the expected duration of incidents with root cause j . The mean time to failure (MTTF) of the component is $1/f_i$, and the expected duration of failure in any given time period is $f_i D_i$.

3.1 Determining RBN structure and parameters

The structure of the RBN and its parameters can be determined through a three-stage process. In the first step, nodes are assigned to each IT component of interest in the model, and arcs are drawn between the nodes wherever a causal relationship is likely. At this stage, the graph need not be acyclic. The causal relationships can be determined from several sources, including the following:

- Logfiles from event monitoring software can be used to deduce correlations and patterns among failure events occurring within a short time window that may indicate causal relationships. System logfiles can be used to provide a real-time view of causal dependencies among events and configuration items (see [2]).
- A Configuration Management Database (CMDB) that stores information on the IT system topology, including relationships among components that can imply causal failure relationships. Causal dependence relations among configuration items can often be inferred from relationships stored in the CMDB, see for example Figure 5.
- Incident records from logfiles as pictured in Figure 2 can provide direct evidence of causal failure relationships from past service record data.

The data from the system logs and service records may be incomplete in several ways. In general, the data will not indicate causal failure relationships explicitly, and this must

be interpreted from the time order and the nature of the events. The time order of events is often not reliable if data have been collected across a system, so in many cases we will not be able to recover statistics on knockdown failures on a component-to-component basis. In these cases, if a real or hypothetical root cause can be established, then we may model the failure effects to be direct consequences of the root cause event. It is important therefore in designing the system logfiles or the help desk processes to ensure that they are configured to accurately collect as much of the relevant data as possible, in order to enable the reliability analysis.

CDM Relationships	Examples	Causal dependence
Y runsOn/basedOn/DeployedTo/storedOn X X contains/provides Y	Unix Y runsOn ComputerSystem X Web App Y deployedTo WebsphereServer X FileSystems Y storedOn FCVolume X FCSwitch X contains FCPort Y	X → Y (deterministic), Y → X (probabilistic)
X controls/manages Y	Controller X controls DiskDrive Y StorageSubSystem X controls StoragePool Y	X → Y (deterministic)
Y uses/connectedTo/communicatesTo/accesses X	App Y uses DB X App Y communicatesTo App X FCPort X connectedTo FCPort Y	X → Y (probabilistic), Y → X (probabilistic)
Y occursBefore X	Service Y occursBefore Service X	X → Y (deterministic)
Y federates X ₁ , ..., X _n	Service Y federates Applications X ₁ , ..., X _n	X ₁ , ..., X _n → Y (deterministic OR)
X ₁ , X ₂ , ..., X _n memberOf Y	WebsphereServers X ₁ , ..., X _n membersOf WebsphereCluster Y CopySet X ₁ , ..., X _n membersOf DataSet Y	X ₁ , ..., X _n → Y (deterministic AND)

Figure 5. Mapping CDMB relationships to causal arcs.

In the second stage, the network parameters are estimated from data. Again, both service records and event monitoring logs can be used to determine failure durations, inherent failure rates, and transfer probabilities. Inherent failure rates are determined by the number of times that a component is the root cause of a given incident within a given time period, such as a day. (In cases where no single root cause can be identified, the first component observed to fail in the incident may be substituted instead, or the number of failures without any observed prior cause.) Transfer probabilities can also be estimated in a similar way, using for example the matrix of affected components per incident pictured in Figure 2, or by the frequency with which a failure event for a component appears correlated with a “root cause” event in an event monitoring log. In each of these cases, there may not be enough data to populate all the inherent failure rates or transfer probabilities through direct estimation; gaps may be filled either by pooling failure data across IT components of similar type, or by incorporating prior knowledge by placing a prior distribution over the parameter space. As these parameters are probabilities taking values in [0,1] estimated from binary data, a typical choice would be to use a conjugate prior such as the Beta distribution to encode this uncertainty [12].

In the third stage, the cycles are removed from the graph. This can be done in several ways. A generic way is to add duplicate nodes to the graph for any set of nodes that

comprise a cycle in the graph. The original and duplicate nodes represent the state of the IT components at two discrete points in time, with the original nodes corresponding to the earlier time point. See Figure 6.

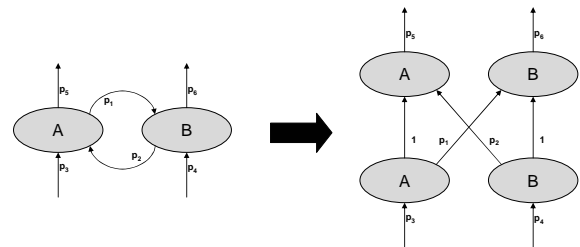


Figure 6. Removing a cycle from a graph.

In certain special cases, a less expensive technique may be used. Consider the Douglas RBN in Figure 3 for example, causal relations were assumed to be deterministic in all “contains” relationships among all the components, so that the transfer probabilities among the upward vertical relationships of machine to image to application to service were all equal to one (i.e., if a machine fails, the image also always fails, etc.) Transfer probabilities for down-chain relationships (e.g., from application to image) could be stochastic, however. By replacing the deterministic arcs going up the chain with arcs that directly connected the IT components in the chain with the top-level service node, the resulting network was acyclic. This replacement does not affect the result of the calculation of the service availability, although adjustments do need to be made to the estimated availability of the intermediate components. In this description, we implicitly assume that failures in the overall system are rare and that failure incidents with distinct root causes do not overlap in time. We also assume that the sum of all the inherent failure rates is (much) less than one, and the probability of all components being up is equal to one minus this sum.

4 Evaluation

Engineers often strive to build and maintain as reliable real-world IT deployments as possible. As a result, while failures may generally be common in a large environment, they are often rare on a per component per dependency basis. Both training and test data are thus scarce. In light of this, we designed and conducted data-light experiments for evaluating the Bayesian network reliability model (RBN) to address the question of whether we can obtain a meaningful model from the limited amount of data we may have. In addition, we also want to understand the correlation between the model uncertainty and the amount of training

data. Since exact analysis of probability propagation in Bayesian networks is difficult ([7]), we relied on a Monte-Carlo approach for these experiments. Each iteration of the Monte-Carlo procedure involved the following steps:

1. Use the Douglas data set to update each beta distribution for the Douglas RBN;
2. Draw a random (inherent/transfer) probability value from each beta distribution in the Douglas RBN;
3. Using the drawn probability values, perform probability propagation to computer service down probabilities.

4.1 Model variance/confidence

Our first experiment studies how the uncertainty of the Douglas RBN would be impacted by the size of training data set. To this end, we drew samples with replacement from the Douglas data set to create new data sets of varying size equivalent to a half-year, one-year, 1.5-year, etc., sample size. In each of 200,000 replications of the Monte Carlo procedure described above, we estimated the failure probability f for each Douglas service. The results of this experiment are depicted in Figure 7.

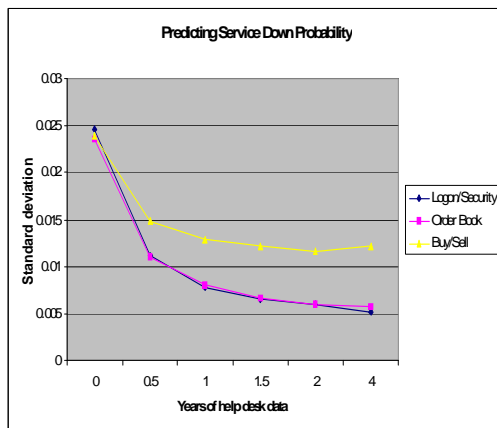


Figure 7. Douglas RBN uncertainty.

As one would expect, regardless of the Douglas service concerned, the uncertainty level decreases as more training data becomes available. It saturates after the point of 2-year help desk data and does not drop to zero, as there are some model parameters for which no data exists in the Douglas data set. The uncertainty associated with the *Buy and Sell* service is always the greatest, which is also intuitive in that it depends on the other two services to function. The

standard deviations of probability assessment when there is no training data (see the data points for 0 year of help desk data) are within 0.025 (the relative standard deviations are within 25% of the mean probability assessment results), acceptable for the purpose of acquiring coarse estimation using the model. We believe this may be attributed to the amount of deterministic dependencies in the Douglas RBN (i.e. those transfer probabilities that are not derived from data), and we investigate this further in the next section.

4.2 The benefits of domain knowledge

In order to validate our earlier assertion that the incorporation of causality domain knowledge has reduced RBN's data sensitivity, we conducted an experiment to compare three different RBNs built for the Douglas excerpt described in Section 2, each leveraging a different amount of domain knowledge. The first of these RBNs is the RBN constructed using the methodology proposed in this paper, and uses the dependencies among Douglas system components to determine its structure as well as quantify some of its transfer probabilities as is detailed in Section 3.1. The second RBN is one where the Douglas topology is used to determine its structure only. All RBN probabilities are computed from the Douglas outage incident data, or set to 0.5 if no data is available. The third RBN is entirely generated from data and no domain knowledge is utilized, that is, a standard Bayesian network structure learning algorithm [7] is first used to determine the RBN structure and the RBN probabilities are subsequently computed as in the second RBN above. We name these three RBNs as RBN[S,P], RBN[S] and RBN[] according to their level of domain knowledge utilization, where S denotes structural knowledge and P denotes probability/parameter knowledge.

Again, 200,000 Monte Carlo iterations were run for each network, and we compared the variance of the estimators for the availability of the Buy and Sell service. Figure 8 illustrates that domain knowledge also increases the relative variance and thus the uncertainty of the models, although uncertainty appears to be less pressing an issue when the model behaving not much differently from a random BN and is not even capable of representing the training data.

5 Related Work

The engineering community have long been using Markov chains [13] as a theoretical base for systems reliability modeling in a variety of domains. However, such models are restricted in terms of scalability and make strong assumptions on exponential failure distributions and memory-less properties [3]. Another popular traditional reliability modeling technique is fault trees [5]. Crucially,

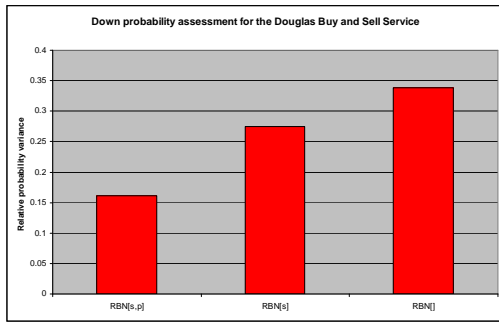


Figure 8. Relative variance of service down prediction by the three different RBNs.

fault trees can not naturally describe common causes of failures, a regular occurrence in complex IT configurations. Specific to the IT domain, system or service topology are traditionally used to estimate availability. Bayesian networks have more recently become the primary tools for analytical reliability and availability modeling. Examples can be drawn from various domains including corporate actions processing [9] and foreign exchange [6], and IT systems [10]. Note that in these approaches, the Bayesian network theorem was treated as a probabilistic, analytical framework. Unlike in this paper, its capacity as a machine learning technique has not been exploited. Manually constructing a Bayesian network for a complex environment can be error-prone and time-consuming.

Availability models fully or partially generated from data have recently started to emerge. The approach described in this paper falls into this category. Murphy et. al [4] attempted to model the availability of a set of servers using a Bayesian network that additionally accounts for failure dependency over time, an interesting consideration that overlaps with our plans for future work. In contrast to the model described in this paper, servers are the only type of elements considered. The structure of their Bayesian network structure is generated entirely from data. While this approach may have the highly attractive advantage of discovering hidden failure dependencies, it may prove computationally too expensive or too data-demanding as the system scales. Challagulla et. al. [1] use both data collected from reliability testing and in operation to predict service availability in service-oriented environments [8]. The resulting model is simply a set of mapping as to how likely the service will give the wrong output given a specific input. Supatgiat et. al. [12] developed a probabilistic model to map low-level IT element availability status to high-level service availability status. A separate mapping is learned from data for every type of IT elements. Their model can be viewed as the Bayesian network model proposed in this paper collapsed into two layers. Chen et. al. [11] applied neural networks to

predict software reliability. A major common drawback of these approaches, however, is their black-box nature, which precludes the incorporation of domain knowledge to allow for quicker and more accurate modeling given limited data.

There is also some recent work on learning performance models from performance logs (e.g. response time models trained in [15]). If one introduces an appropriate out-of-patience threshold (beyond which point the users will concede there would not be an response), these models can shed light on IT systems's availability from a different angle than the availability models outlined in the previous subsection.

6 Discussions and Future Work

Two immediate extensions to this work are (i) using multivariate or continuous variables to represent factors such as the workload on a server, and (ii) adopting a more powerful probabilistic framework like *Dynamic Bayesian Network (DBN)* to capture the time-varying aspect of IT availability. In the longer term, developing a systematic way to meaningfully merge data collected across deployments would allow us more data to learn accurate models from. A key problem related to this is the problem of incorporating data that were collected from out-of-date system configurations. In some cases, weights can be assigned to the data to favor more recent events, but in other cases it may be necessary to exclude certain data. In previous work, this was addressed in an ad-hoc way; in future work we hope to provide more reliable guidance to the use of data from different systems or from different epochs.

References

- [1] V. U. B. Challagulla, F. B. Bastani, R. A. Paul, W.-T. Tsai, and Y. Chen. A machine learning-based reliability assessment model for critical software systems. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference - Vol. 1- (COMPSAC 2007)*, pages 79–86, Washington, DC, USA, 2007. IEEE Computer Society.
- [2] R. Chang, F. Bernardini, E. Cope, C. Perng, E. So, C. Tang, and T. Tao. MITRIS: Proactive IT operational risk management using temporal relationship discovery technologies. In *Poster Proceedings of the IBM Academy Of Technology Conference on Proactive Problem Prediction, Avoidance, and Diagnosis (P3AD)*, April 2009.
- [3] K. M. Greenan and J. J. Wylie. Reliability Markov models are becoming unreliable. In *Posters of 6th USENIX Conference on File and Storage Technologies (FAST '08)*. USENIX Association, 2008.
- [4] R. Herbrich, T. Graepel, and B. Murphy. Structure from failure. In *Proceedings of 2005 Annual Conference of National Information Processing Symposium (NIPS'05)*, 2005.
- [5] IEC. Fault tree analysis. In *IEC 61025*. IEC, 1990.
- [6] G. Kumasi. Operational risk management: Implementing a Bayesian network for foreign exchange and money market settlement.
- [7] R. Neapolitan. *Probabilistic Reasoning in Expert Systems: Theories and Algorithms*. Wiley-Interscience, 1990.

- [8] M. P. Papazoglou and D. Georgakopoulos. Introduction of service-oriented computing. *Commun. ACM*, 46(10):24–28, 2003.
- [9] S. Ramamurthy and H. Arora. Operational risk and probabilistic networks: An application to corporate actions processing. Technical report, Infosys Technologies, November 2005.
- [10] R. Roshandel, N. Medvidovic, and L. Golubchik. A Bayesian model for predicting reliability of software systems at the architectural level. pages 108–126. Springer Berlin / Heidelberg, 2008.
- [11] Y.-S. Su and C.-Y. Huang. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *J. Syst. Softw.*, 80(4):606–615, 2007.
- [12] C. Supatgiat and C. Berrospi-Ramis. Predicting a new risk level of an information technology system after a configuration change. Working Paper, IBM Zurich Research Lab, May 2006.
- [13] A. Villemeur. *Reliability, availability, maintainability and safety assessment: methods and techniques*. Wiley, 1992.
- [14] E. Wustenhoff. Service level agreement in the data center. Technical report, Sun Microsystems, April 2002.
- [15] S. Zhang, I. Cohen, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 644–653, Washington, DC, USA, 2005. IEEE Computer Society.