Model-Based Evaluation of System Resilience

J. F. Meyer

jfm@umich.edu

Department of Electrical Engineering and Computer Science
Division of Computer Science and Engineering
University of Michigan, Ann Arbor, Michigan USA

***Abstract*—The notion of system resilience is receiving increased attention in domains ranging from safety-critical applications to ubiquitous computing. After reviewing how resilience has been defined in various contexts, we focus on a dependability-based definition and a performability-based extension thereof. Modeling problems posed by the quantitative evaluation of system resilience are then discussed, along with some suggestions as to how they might be solved.**

I. Introduction

Scientific investigations concerning the resilience of large, natural systems date back to the mid-1970s, typified by the pioneering work of Holling on the resilience and stability of complex ecological systems [1]. In the more specific context of computer, communication, and control systems, the term *resilient* has served as a roughly defined synonym for "fault-tolerant," again dating back to the mid-70s. However, as noted by Laprie [2], it was given a more specific meaning in the preface of a 1985 collection of papers edited by Anderson [3], namely the addition of "robustness" as a key attribute, i.e., "the ability of a system to deliver service under conditions that lie beyond its normal domain of operation." In effect, this extended usual concerns regarding the tolerance of anticipated faults to include unanticipated conditions/changes that a system may face, especially over long periods of utilization.

During the past decade, *system resilience* has received increased attention due to research efforts in several system domains. Examples include multi-partner projects such as IRIS (Infrastructure for Resilient Internet Systems [4]) in the United States and ReSIST (Resilience for Survivability in IST [5]) in Europe. Since 2008, the High Performance Computing (HPC) community has held annual workshops on *Resiliency in HPC*, where the 2012 workshop focused on the resilience of clusters, clouds, and grids [6]. In the domain of safety-critical systems, *resilience engineering* has emerged as a means of actively anticipating changes in risk prior to the occurrence of resulting damage [7], [8]. Investigations in this regard have been broadened to include industrial, ecological, and social systems, as exemplified by work at Ohio State University's Center for Resilience [9]. Most recently, the United States Department of Defense has initiated a science and technology priority area referred to as "Engineered Resilient Systems," aimed at meeting the challenges of todays dynamic, uncertain defense environment with advanced engineering [10].

The sections that follow expand on a talk [11] presented at the *9th Int'l Workshop on Performability Modeling of Computer and Communication Systems* (PMCCS-9) in 2009. We begin by reviewing certain definitions of resilience which have been proposed in conjunction with the investigations noted above. In turn, we show how the ReSIST definition can be stated more generally in terms of the concept of *performability* [12], [13], thereby accounting for properties such as degradable performance. Modeling problems posed by the quantitative evaluation of resilience (whether it be dependability-based or performability-based) are then discussed, along with some suggestions as to how they might be solved. This includes a proposed

two-fold application of a modeling technique familiar to performability evaluation.

II. Resilience

Contemporary definitions of system resilience differ somewhat according to the assumed nature of a system's application environment. A common property, however, is the ability to cope with unanticipated system and environmental conditions that might otherwise cause a loss of acceptable service (failure).

For example, in the context of applications where safety is the principal concern (particularly human safety, where failures can result in the loss of lives), Hollnagel (in the Prologue of [8]) defines resilience as:

> The intrinsic ability of a system to adjust its functioning prior to, during, or following changes and disturbances, so that it can sustain required operations under both expected and unexpected conditions.

Note that the "under ... unexpected conditions" qualification of this definition is similar to the "robustness" aspect of being resilient, per the characterization in the preface of [3].

The US DoD "Engineered Resilient Systems" initiative defines a resilient system to be one which

> ... is trusted and effective out of the box in a wide range of contexts, easily adapted to many others through reconfiguration or replacement, with graceful and detectable degradation of function.

Of particular relevance to the discussion in Section III is the reference to "graceful ... degradation of function" in the DoD definition. Indeed, in addition to degradable performance being a desired property of resilient systems, we contend that there's a need to account for this property when defining, measuring, and evaluating system resilience.

With respect to highly-distributed applications such as ubiquitous (pervasive) computing, the ReSIST project cited earlier has devoted considerable work to defining resilience and relating it to the notion of *dependability* [14]. Here, the targeted systems are large, networked information infrastructures, referred to simply as *ubiquitous systems*. The following development of the ReSIST definition of resilience is quoted verbatim from the Laprie paper cited earlier [2, page G-8].

> With such ubiquitous systems, what is at stake is to maintain dependability, i.e., the ability to deliver service that can justifiably be trusted in spite of continuous changes. Our definition of resilience is then:
>
> > The persistence of service delivery that can justifiably be trusted, when facing changes.
>
> The definition given above builds on the initial definition of dependability, which emphasizes justifiably trusted service. In a similar spirit, the alternate definition of dependability, which emphasizes the avoidance of unacceptably frequent or severe failures, could be used, leading to an alternate definition of resilience:
>
> > The persistence of the avoidance of failures that are unacceptably frequent or severe, when facing changes.
>
> From what precedes, it appears clearly that a shorthand definition of resilience is:
>
> > The persistence of dependability when facing changes.

Although concern with unanticipated conditions is not explicit in the above definition, it becomes obvious once "changes" are defined in various ReSIST documents. In particular, they introduce a "prospect" dimension of change that includes an *unforeseen* category: see Figure 1 (taken from [2, page G-9]).

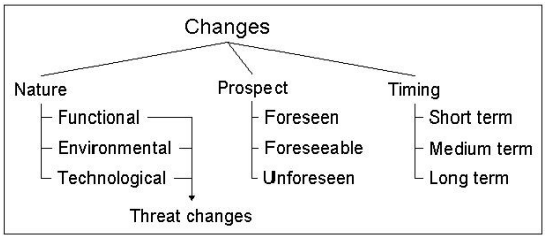

Fig. 1. ReSIST Classification of Changes

These three definitions therefore share the common property we noted at the outset, namely the ability to tolerate unanticipated system and environmental changes that might otherwise cause a loss of acceptable service. However, this prompts the following question. Rather than complicate things by introducing new terms (resilience, change) into the dependability-related vocabulary, why not regard such changes as yet another fault class? (Note that the *foreseen* and *foreseeable* changes of the ReSIST classification comfortably refer to more traditional fault types.) As a consequence, resilience would become synonymous with dependability and change tolerance would reduce to fault tolerance. Indeed, concern with unanticipated phenomena by the fault-tolerant computing community dates back to an IEEE Workshop on "Designing for the Unexpected," held in St. Thomas, Virgin Islands 35 years ago (1978).

On the other hand, introduction of the term "resilience" in a dependable systems context serves to signal the fact that additional types of change are being accounted for (the motivation for Anderson's "robust" qualification). Moreover, changes need not have the negative connotation that typically accompanies the term "fault." For example, is a change in the use environment of an enterprise system, resulting say from its sale to another company, a negative thing? But perhaps the best reason for introducing this new terminology is the fact that the current classification of fault types, e.g., as described in [14] (see Fig. 2) is sufficiently complicated to discourage further elaboration.

Finally, there are a couple of options regarding the semantics of "change" in the ReSIST definition of resilience. The first restricts its meaning to conditions that lie outside of the fault types described in Fig. 2. Concern with more traditional fault classes is then implied by the term "dependability." A second option is that "change" has a more general meaning which includes "fault" as a special case. The latter is a more logical choice (just as rectangles include squares) and, according to responsible sources on the ReSIST team, this is indeed the intended interpretation.

There are other variations on the resilience theme that could be likewise be reviewed, e.g., see the introductory sections of [2], [15]. However, the above should serve as adequate background for the purposes noted at the end of Section I.

III. Performability-Based Resilience

Let us now examine how the ReSIST definition can be generalized so as to account for degradations in performance, particularly when performance is identified with the quality of delivered services.

A. Extending the Definition

Regarding first the definition aspect, the notions described in the previous section are mainly "success-oriented" in that they stress the persistence of correct service delivery in the presence of disruptions/changes. ("Service failure" is identified with a transition from correct to incorrect delivery; see [14, Sec. 2.2], for example.) In this sense, they are "dependability-based" (an obvious qualification for the ReSIST definition).

In the case of safety-critical systems, this focus is justifiable due to the catastrophic consequences of failure. Failure-free service deliveries are therefore the paramount concern. However, in the more general context of ubiquitous systems, this focus is narrower than need be. Instead, the notion of resilience can be extended just as measures of performability extend measures of dependability (e.g., reliability and availability). In particular, a performability measure has several interesting properties that bode well for resilience evaluation.

1) It is able to account for dynamics of system structure and behavior that affect both performance (in the strict sense) and dependability.

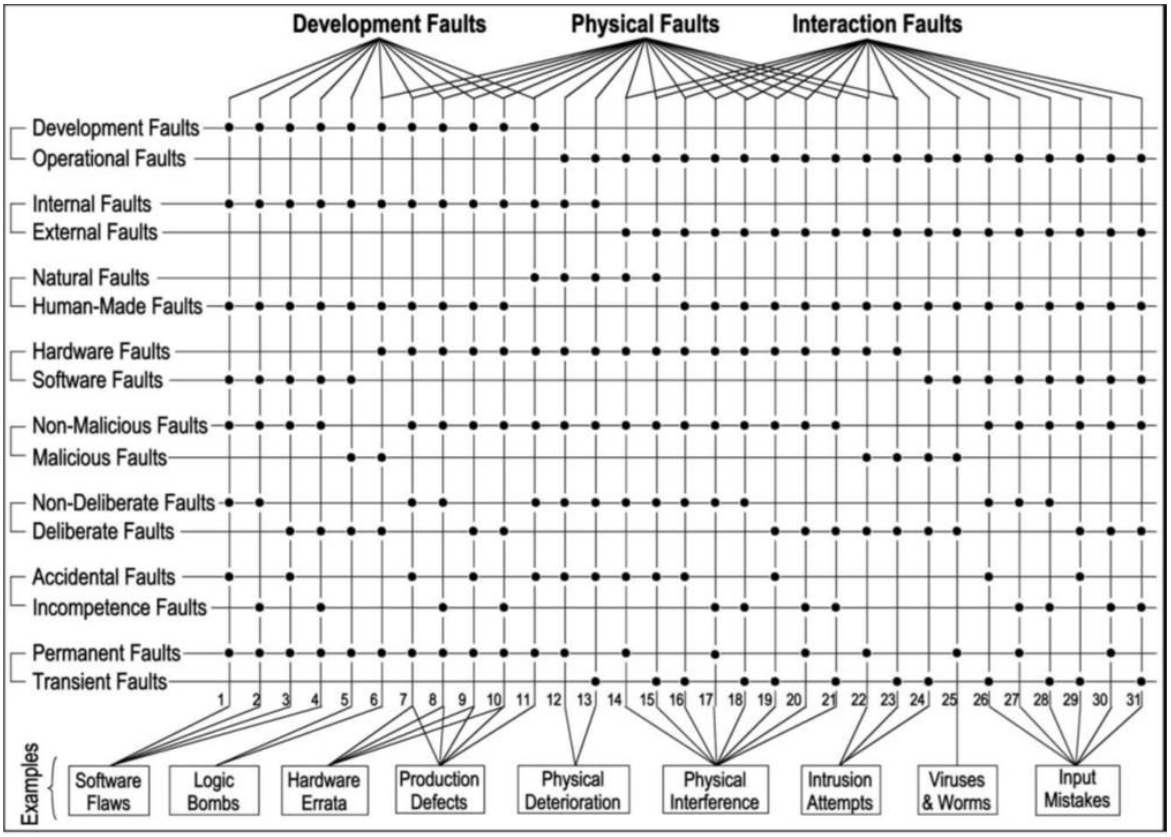

Fig. 2. Fault Classes

2) In particular, it can account for degradations in performance (including high-level views thereof such as service quality) that lie above the threshold of system failure.
3) It is able to unify performance and dependability aspects by expressing accomplishment in terms of one-dimensional values (typically real numbers).
4) Its values can depend on both what a system is and what it does throughout a specified period of utilization.

Hence, when expressed in the form of the shorthand version of the ReSIST definition, we have:

Def.: *Resilience* is the persistence of performability when facing changes.

Stated informally, a performability measure quantifies a system's ability to perform in the face of faults. Accordingly. this extended notion of resilience can viewed as a system's ability to perform in the face of changes. Based on the features noted above, this opens doors that are closed to a dependability-based definition.

IV. Resilience Evaluation

Although considerable effort has been devoted to conceptual aspects of resilience and how they relate to notions of adaptivity, stability, evolvability, etc., little work has been done regarding how resilience is to be measured quantitatively and, in turn, how systems can be evaluated with respect to such measures. A notable initial undertaking in this

regard is the work of Trivedi, et al. [15]. Using some well-known dependability/performability measures and models (continuous-time Markov chains), infrequent changes are represented by varying the values of certain model parameters. The results are interesting, but relatively elementary in nature. As the authors suggest in their concluding remarks,"Other types of stochastic models can also be used to study resilience of various systems."

It's certainly the case that the specific choice of a measure will vary according to the type of system being evaluated (the object system) and the nature of its application environment. However, it should be possible to pursue some strategies for quantitative resilience evaluation that are relatively independent of the *total system* (object system plus environment) in question.

Regarding the kinds of change to be dealt with, the domain of fault-types considered in both dependability and performability evaluation has expanded considerably over the past 35 years (again see Fig. 2). However, the term "fault" continues to refer mainly to anticipated (foreseen, foreseeable) changes in a system or its environment. Hence, for either of the two x-based definitions of resilience (x = dependability or performability) discussed in the previous section, the principal added ingredient is the persistence of x with respect to *unanticipated changes* (UCs), i.e., ones that are "unforeseen" according to the ReSIST classification (Fig. 1).

Accordingly, accommodating UCs when evaluating resilience involves the consideration of system/environment dynamics that are beyond those typically addressed in the evaluation of x. In particular, they include evolutionary changes in the system's environment that occur more slowly over longer periods of system use. They also include adaptive changes in system structure and behavior that respond to environment changes, thereby permitting x to persist.

Quantitative resilience evaluation thus poses a number of interesting challenges. Moreover, doing it by observing measure values obtained from an actual system (measurement-based evaluation) will be extremely difficult. (How long must someone wait for the next unanticipated change to occur?) Accordingly, model-based evaluation, either with analytic models or simulation models, appears to be the more practicable option for obtaining quantitative results. This is not to say that observed data concerning system resilience is useless. To the contrary, real-world observations concerning UCs and their consequences will be of utmost importance in the construction of resilience models (e.g., determining values of certain model parameters).

In particular, these challenges include the following tasks.

T1) Specifying resilience measures in high-level, user-oriented terms.

T2) Constructing total system models that
   a) permit lower-level formulations of the specified measures, and
   b) account for the additional dynamics of UCs.

T3) evaluating the measures of T1 based on the models of T2 (resilience model solution).

Among these, task T1 may be the easiest to accomplish. For example, given a particular type of object system and use environment, one can select a favorite quantitative dependability or performability measure and then specify what is meant by "persistence." For example, if "to persist" is "to exist" then the resulting resilience measure coincides with the underlying x measure, except that now it can reflect effects of changes such as UCs (as well as faults). However, other interpretations can lead to other measures. For instance, suppose "persist" has the meaning of "holding on" to some acceptable level of ability to serve, e.g., stay at or above some lower bound b on the mean service quality (MSQ). Resilience in this case is then captured by the performability measure:

> Fraction of some specified utilization period wherein MSQ $\geq b$.

Tasks T2 and T3 are much more challenging. Regarding T2, one must be able, in some way, to represent the probabilistic nature of UCs and relate these dynamics to the specified resilience measure. It appears likely that the origin of UCs will be primarily external (will lie outside the boundary of the object system in question). This is due to the fact

that internal changes, being confined to the system itself, are typically better understood and therefore more likely to be anticipated. However, depending on how the boundary of the object system is defined, some UCs may be internal.

External UCs, on the other hand, can have global and even extraterrestrial origins, e.g., solar radiation bursts and meteor impacts. Accordingly, the representation of UCs will reside primarily in the environment part of a total system model.

The temporal nature of a UC can be broadly classified as being either *discrete* (a DUC) or *continuous* (a CUC), where the two are distinguished more precisely as follows. A DUC has a specific time of occurrence (is an event) and is likely to occur infrequently. (If a change occurs frequently then it should be observed relatively often and, hence, should be anticipated; in this case the change is a fault.) On the other hand, a CUC evolves without having a perceptible occurrence time. Moreover, it is likely to evolve slowly since rapidly evolving changes are more easily observed and again can be anticipated by system designers.

Some stochastic implications of the above distinctions are as follows. For a DUC, the mean time between its occurrences (or to the only occurrence if it's a one-off event) is much longer compared with mean times between fault occurrences. Hence, DUC occurrence probabilities, even during lengthy utilization periods, will likely be extremely low. For CUCs, a continuous state space may be required in order to represent slowly evolving changes.

A resilient system should be able to tolerate UCs, even though such changes are not anticipated. At this juncture, it is unclear as to just how the tolerance of UCs (and other non-fault changes) will differ from methods of tolerating faults. One difference may be the ability to adapt to slowly evolving changes. For example, unanticipated growth in demands on a server farm can cause degradations in mean service quality that could eventually become unacceptable. Tolerance of such a CUC might then depend on just how the servers are interconnected. In particular, if their interconnection permits the server pool to be expanded without service interruptions then this CUC can be tolerated by so increasing the system's capacity to serve.

Finally, regarding model-based solution methods for quantitative resilience measures (task T3), these appear to be more difficult when compared to techniques employed for typical dependability and performability models (those in which the represented changes are restricted to faults). One rather obvious reason for this is the need to account for the effects of UCs with properties of the type discussed above. In particular, infrequently occurring DUCs will likely have a time granularity which exceeds that of typical fault occurrences by many orders of magnitude.

In the case of performability-based resilience measures, this suggests a model-based evaluation method which, with a considerable investment of energy and time, just might work. We call it "Courtois Revisited" since it utilizes a popular performability solution technique [16] based on Courtois' theory of "near complete decomposability" [17]. It relies on the underlying assumption that frequently occurring events (e.g., service related events) are likely to approach steady-state behavior between occurrences of events having much larger mean inter-arrival times (e.g., faults and fault recoveries).

So why not employ a second-pass Courtois-type decomposition in order to accommodate DUCs? More precisely, in the second pass, faults and fault recoveries become the frequently occurring events relative to the much less frequent occurrences of DUCs. The proposed model-based resilience evaluation method then proceeds as follows.

S1) Postulate a set of system-environment states representing effects of DUCs (the "DUC-states").

S2) Evaluate a "performability rate" for each such state, e.g., the steady-state mean service quality, via usual methods of steady-state performability evaluation.

S3) Conduct another performability evaluation relative to the DUC dynamics, where the reward rates assigned to DUC-states correspond with the results obtained in step S2.

With respect to a given DUC-state, the evaluation in step S2 acounts for a system's ability to perform in the face of fault-related changes (fault

occurrences, fault recoveries) which, by assumption, occur much more frequently than DUCs. It is also assumed that the fault dynamics are such that a steady-state performability distribution exists for each S2 evaluation.

In step S3, on the other hand, a steady-state assumption makes no sense, due to the highly infrequent occurrences of DUCs. Even if the system's utilization period is very long (say 100 years), the mean time to an occurrence of a DUC could be of the same order of magnitude or even larger. Accordingly, step S3 calls for the evaluation of performability (e.g., the distribution of accumulated reward) over a finite time period using sophisticated transient solution methods of the type implemented in Möbius [18], [19].

Step S1 appears to be the most daunting if the model is to reflect reality. Among other things, it will require an extensive study of how DUCs occur in actual systems. Short of that, one can postulate more idealized DUC assumptions and perhaps obtain some "meaningful relationships" regarding how they impact performability-based resilience.

V. Summary

So what has this investigation accomplished? It reviewed several notions of resilience, proposed a performability extension thereof, and put some syntactic meat on the semantic bones of unanticipated changes. It then addressed some important issues regarding quantitative, model-based resilience evaluation, concluding with a proposed 2xCourtois modeling method for performability-based resilience measures.

References

[1] C. S. Holling, "Resilience and stability of ecological systems," *Annual Review of Ecology and Systematics*, vol. 4, no. 1, pp. 1–23, 1973.

[2] J.-C. Laprie, "From dependability to resilience," in *Proc. IEEE Int. Conf. on Dependable Systems and Networks*, vol. Supplemental, 2008, pp. G8–G9.

[3] T. Anderson, Ed., *Resilient Computing Systems*. Collins, 1985.

[4] http://www.iris.lcs.mit.edu.

[5] http://www.resist-noe.org.

[6] http://xcr.cenit.latech.edu/resilience2012.

[7] E. Hollnagel, D. D. Woods, and N. Leveson, Eds., *Resilience Engineering - Concepts and Precepts*. Ashcroft, 2006.

[8] E. Hollnagel, J. Paries, D. D. Woods, and J. Wreathall, Eds., *Resilience Engineering in Practice: A Guidebook*. Ashcroft, 2011.

[9] http://www.resilience.osu.edu.

[10] http://www.acq.osd.mil/chieftechnologist/publications.

[11] J. F. Meyer, "Defining and evaluating resilience: A performability perspective (extended abstract)," in *Proc. 9th Int'l Workshop on the Modeling of Computer and Communication Systems*, September 2009.

[12] ——, "On evaluating the performability of degradable computing systems," *IEEE Trans. Computers*, vol. C-29, no. 8, pp. 720–731, August 1980.

[13] B. R. Haverkort, R. Marie, G. Rubino, and K. Trivedi, Eds., *Performability Modelling: Techniques and Tools*. Wiley, 2001.

[14] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[15] K. Trivedi, D. S. Kim, and R. Ghosh, "Resilience in computer systems and networks," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, Nov., pp. 74–77.

[16] J. F. Meyer, "Closed-form solutions of performability," *IEEE Transactions on Compuers*, vol. C-31, no. 7, pp. 648–657, July 1982.

[17] P.-J. Courtois, *Decomposability: Queueing and Computer System Applications*. Academic Press, 1977.

[18] M. Qureshi and W. Sanders, "A new methodology for calculating distributions of reward accumulated during a finite interval," in *Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on*, 1996, pp. 116–125.

[19] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. Doyle, W. Sanders, and P. Webster, "The Möbius framework and its implementation," *Software Engineering, IEEE Transactions on*, vol. 28, no. 10, pp. 956–969, 2002.