Complexity in Engineering: The Silent Killer

Day W. Radebaugh
Assistant Professor of Philosophy
Department of Philosophy
Wichita State University
Wichita, Kansas 67260

Abstract

A review of the list of recent technological disasters suggests that the risk to society of large-scale engineering projects has grown in proportion to the complexity of the designed system. Illustrative cases include the Challenger explosion, the power grid blackouts of the Northeast United States, the Chernobyl nuclear explosion, and a host of others. Even worse, system complexity renders the process of diagnosing and correcting these failures more difficult still.

If complexity of a system increases its potential risk to failure, then it would make sense to develop a robust measure of engineering complexity, and to teach engineers the methods that could be used to assess the complexity of a system that they are considering. In fact, one could argue that an assessment of system complexity should be done prior to any consideration of the ethical implications of a system, using the common-sense principle that a designer cannot evaluate ethical consequences if he cannot reliably predict the behavior of the project. An appreciation of system complexity deserves a prominent place in the engineering ethics curriculum.

However, when one searches the literature for a definite and useful measure of engineering complexity, one finds instead a set of measures that are inconsistent and only partially relevant. This paper will review and evaluate the most common measures of complexity that might be applicable to the engineering domain. Following an assessment of the applicability of these measures, the author will suggest a more promising approach that will provide engineers a useful guide to assessing system complexity.

I. The Nature of the Concern

Engineering implementations during the past several decades at least have been characterized by increasing complexity. This complexity, at first glance, can be described in two dimensions. The first dimension could be categorized as complexity of inherent design, and describes the underlying processes responsible for the system's behavior. The second dimension is perhaps more generally familiar, and refers to the complexity of usage or perceived complexity: the nature of the interaction of the system with its environment, either other systems or the human user itself. The reader can quite likely supply examples of interactions with everyday systems that illustrate this occasionally bewildering complexity. Part of the reason for Apple's remarkable success in introducing the iPod and iPhone can be attributed to its unique simplicity of usage.

Both types of complexity make it more difficult to understand modern technological systems, and reduce our subsequent ability to control their behavior. Such complexity is at least partly responsible for a number of recent engineering disasters, such as the massive outages of the electrical power grid in 1965 and later in 2003, the nuclear accident at Three Mile Island in 1979, and the various disruptions of the World Wide Web. For example, in the Three Mile Island disaster, engineers were unable to understand and diagnose the behavior of the reactor, which almost led to its destruction. During the power blackouts, engineers were similarly unable to understand what had happened in the highly complex power grid, and so were unable to prevent the outages.

Yet the growth of complexity, as a trend in its own right, goes largely unremarked. Complexity can indeed be called the "silent killer" of engineering, as high blood was described in earlier times: a condition which seems to grow imperceptibly worse, and whose consequences in the extreme are sudden and catastrophic.

Yet the issue of complexity, per se, warrants special attention. The author would agree with Ulrich Wengenroth[1], who states that

> …in fostering new and ever more complex technology, engineers have substantially if unintentionally created a dynamic environment in which the number of problems is constantly expanding. Rather than just reducing complexity and uncertainty by scientific research, engineers use all available knowledge to increase complexity (and therefore inevitably uncertainty) in the man-made world. It is therefore wise to develop strategies to manage engineering complexity in the absence of complete knowledge….Managing engineering complexity is not just a cognitive challenge; it is also an ethical issue…

Mitchum and Siang[2] capture the ethical nature of this concern when they say

> Forget the debate about utilitarian ethics, deontological ethics, or virtue ethics, we are losing moral agency in our growing collective inability to predict the consequences of complex systems…

In summary, while much attention is devoted to the proper method of teaching ethical sensitivity to engineering students, it is perhaps more important to ask how we can evaluate the ethical consequences of technological systems in the first place, if we cannot adequately understand their complex behavior.

It is appropriate to ask why engineering systems have become so complex. It seems self evident that complex functionality is required to solve the increasing number of technological problems society confronts. For instance, it is difficult to imagine the modern commercial air transport system without a highly complex air traffic control infrastructure; the nature of the problem requires a complex solution. Yet there is an aspect of engineering complexity which often seems to exceed any reasonable set of functional requirements, and exists for its own sake. In other words, complexity of design seems to thrive on itself, and grow beyond ordinary boundaries. Engineers, like other professionals, will respond to their environment, and will design increasingly complex systems because other designers are doing so. Why is this the case? Even though a designer may want to adopt a simpler solution to a set of requirements, if the system being designed must interact with another very complex system, one will be forced to implement a complex set of interactions to be compatible. In addition, there can be distinct economic incentives to incorporate complexity in design; increased features can establish a marketing advantage and certainly contribute to profit. There seems to be a trend in automotive technology, for example, to introduce complicated controls for various user functions, such as windows, radio, navigation, air conditioning, etc. The same trend can be found in consumer appliances and electronic devices. Further, the profit from complexity can be secured with little additional development cost. Relatively inexpensive software, ubiquitous and cheap sensors and servos can yield highly complex functionality, if desired.

II. Attempts to Characterize and Quantify Complexity

Given its contribution to technological risk, the development of useful characterizations, quantifications, and methods of managing such complexity would seem to be a pressing engineering concern. Yet a scan of the literature reveals a confusing set of definitions and measures of complexity which offer no useful and practical contribution to the management of this problem. As a first attempt, we note that the dictionary[3] defines complexity as "…a whole made up of complicated or interrelated parts…", yet even this intuitive definition does not pass muster. Keeping in mind the dimension of perceived complexity introduced above, we observe that there are many types of processes which can be described in terms of simple processes, yet yield very complex behavior as witnessed by humans who interact with such a system. Examples include chess games, whose rules can be described in very simple terms, or fractal processes, which can be simply described in functional terms but which produce patterns which appear to be very complicated indeed.

Similarly, does complexity imply a mechanism with a lot of parts? Certainly not; a watch contains many parts, but the key fact is that the parts of a watch can be understood in isolation. Given that the functional organization of a watch's subsystems is generally

hierarchical, one can in turn understand the whole operation by understanding the operation of each system in itself, and synthesizing the operation of the whole from the set of predictable interactions among the components.

Complex systems exhibit a number of behaviors that seem to tax our ordinary descriptive classifications. If all engineered systems could be functionally analyzed in a hierarchical fashion, they would be somewhat easier to understand and predict. However, some of the most interesting and challenging cases are non-hierarchical and distributive in nature, where interaction and control is distributed among nodes, thus rendering the system more difficult to model and predict. The behavior of the Internet is such a case. Is it safe to say that complex systems are non-linear in nature? While it is certainly true that the rapid transition of behavior characteristic of non-linear processes can lead to catastrophic consequences, this is by no means universally true. Many engineering disasters can be attributed to the fact that a complex array of components sometimes act in linear, but hard to understand, ways. By the same token, to characterize complex systems as stochastic, or probabilistic in nature is to simply restate the problem, and does not help achieve a practical measure of design complexity useful to the working engineer.

Complex systems, in the most extreme cases, appear to be a new breed, and one is forced to describe their behavior with terms such as "holistic", which in the ordinary sense means that the behavior of the whole is more than the sum of its parts. Analysis of such systems can be a vicious circle, in which the components cannot be completely understood in isolation, and the behavior of the whole cannot reasonably be predicted from its constituents. To say that a system is holistic is simply to defer the problem of analyzing its behavior. By the same token, we tend to say that some complex systems appear to be adaptive, and self-organizing, but once again this leaves us no closer to realizing a useful and practical measure of complexity.

Clearly such one-dimensional qualitative descriptions of complexity do not provide a useful measure of such a process. The most serious quantitative measures of complexity have come from information theory. The most well-known of such measures is called the Algorithmic Information Complexity, and is due to Kolmogorov and Chaitin. AIC is simply the length of the shortest program required to produce a particular output. While such a measure can be straightforwardly quantified, it is difficult to see its relevance to the problem of providing the working engineer with a measure of complexity, with the possible exception to the field of software design. The notion of entropy H from classical information theory

$$H(M) = - \Sigma_m p(m) \log_2 p(m)$$

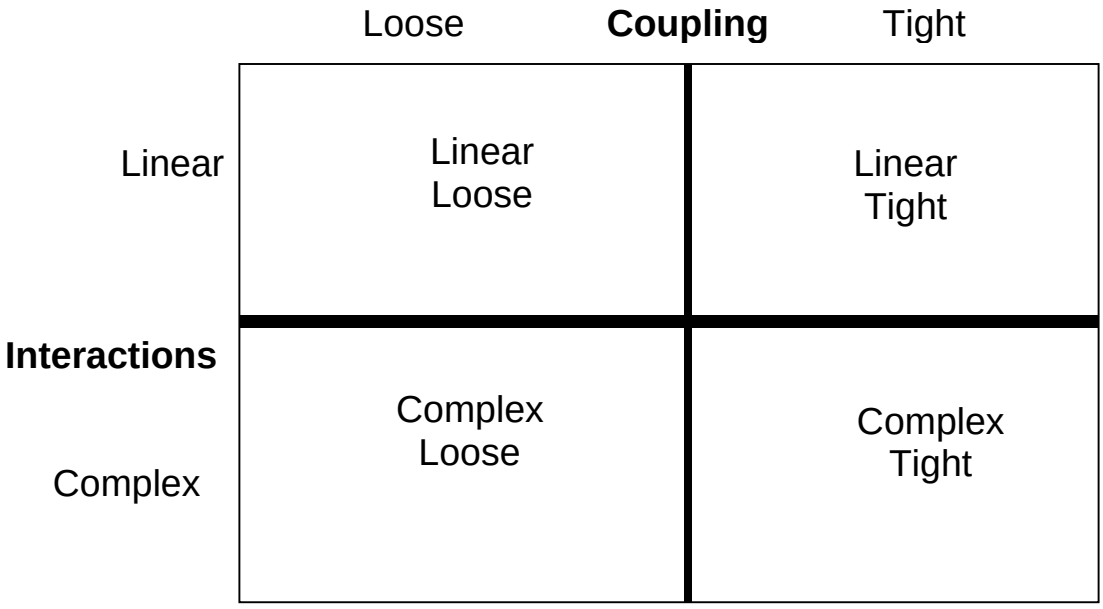
where the information or *surprise* value of a message m = -log p(m), given p(m) is the probability that message m is chosen from the universe of messages in M. As was the case with algorithmic complexity, we must wonder about the relevance of this measure to the problem of estimating the complexity of engineering designs.

III. A More Useful Approach to a Practical Measure of Engineering Complexity

A more useful measure of engineering complexity must provide more than qualitative descriptions, and furthermore be conceptually more relevant than information-theoretical quantifications. A useful measure of complexity must capture the structural dimension of the problem, in the sense that it describes the components of a system and their interactions. Such a measure will at least be intuitively more useful to the engineer, who often visualizes a system in terms of diagrams.
We find the beginnings of such measures in two areas. The first is graph theory, in which system states and relationships are analyzed as graph nodes and edges. We find a quantitative measure in the number of minimal spanning trees of a graph, which indicates how many trees are required to minimally completely span the nodes of a graph. However, the concept of trees implies an hierarchical organization of nodes, but as previously suggested, some of the most irreducibly complex systems are non-hierarchical in nature. Instead, the cyclomatic number, which is an indicator of the number of independent cycles or closed paths in a graph, better captures the idea of non-hierarchical organization.

Can we say more then about the structural nature of components which are non-hierarchically related? Charles Perrow[4] offers the most useful and practical attempt thus far to intuitively characterize complexity among component systems. Perrow classifies complexity in two related dimensions, those of interaction vs. coupling. The dimension of interaction describes, if you will, the complexity that one would perceive looking at a diagram or flowchart of the system's components, and can be quantified in terms of the number of branching paths, feedback loops, and jumps from one sequence of interaction to another. A system with complex interactions has many branching paths, feedback loops, and jumps from one sequence to another. In other words, its system diagram looks a bit like spaghetti. In contrast, a system with linear interactions has very predictable and sequential behavior, and is clearly more easily understood. Coupling is the second dimension of complexity, which attempts to capture the dynamic nature of the interaction among components. Loose coupling is characterized by processing delays, more degrees of freedom, and interactions in which the order of operations can be changed. Tight coupling does not permit such alternatives; there is little or no delay of interaction among components.

Using these categories, we can classify systems in terms of the following matrix:



Perrow's classification system offers a more promising approach to the problem of assessment of complexity of systems. It provides a system that is both roughly quantitative, with 4 categories of relationships, and conceptually intuitive to use.

Perrow offers several examples of the use of his rubric to classify technological systems. For instance, he places nuclear power plants in the complex-tight quadrant. Such systems are complex in the sense that they are controlled with indirect information, using multiple and interacting controls, have feedback loops, and common failure modes. [5] They are tightly coupled insofar as delays in control are generally not possible; that is, events occur very rapidly in sequence.[6] Such a classification should be a useful warning to engineers that there is a significant potential for catastrophic behavior. In response, engineers should examine their design, to determine if some of the complexity can be eliminated. At the very least, these aspects should give an indication of where problems would be likely to occur in nuclear reactors, and provide an indication of what problem scenarios would be most useful to model and practice.

IV. Conclusion

This paper has argued that complexity of engineering design significantly increases the degree of technological risk of a system. The assessment of design complexity is at least as important, and certainly logically prior to, any evaluation of the ethical implications of a technology. A variety of attempts to describe and quantify complexity have been discussed, and a suggestion of a useful type of model, based on Perrow's work, has been proposed. Finally, it is recommended that assessment of complexity be included as a component in the engineering education curriculum.

Bibliographic Information

1. Wengenroth, Ulrich, "Managing engineering complexity – a historical perspective", proposal for The Engineering Systems Symposium at MIT, 2003-2004.

2. Mitchum, Carl and Siang, Sanyin, "Complexity, Simulations and Ethics", Professional Ethics Report, XIV (1), Winter 2001.

3. Merriam-Webster's Collegiate Dictionary, Merriam-Webster, Springfield, MA, 1993.

4. Perrow, Charles, Normal Accidents, Princeton: Princeton Univ. Press, 1999.

5. Perrow, op. cit., p. 88.

6. Perrow, op. cit., p. 96.

Biographical Information

DAY RADEBAUGH is a Visiting Assistant Professor in the Department of Philosophy at Wichita State. He has been at Wichita State since 2003, and has taught courses in Computer Ethics, Engineering Ethics, Introductory Philosophy, Assembly Language Programming for the PC (Computer Science Dept), and Military History (History Dept).