Applied Ergonomics 59 (2017) 581–591

Contents lists available at ScienceDirect

Applied Ergonomics

journal homepage: www.elsevier.com/locate/apergo

Rasmussen's legacy: A paradigm change in engineering for safety

Nancy G. Leveson

Aeronautics and Astronautics Dept., MIT, United States

ARTICLE INFO

Article history:
Received 31 March 2015
Received in revised form
11 October 2015
Accepted 25 January 2016
Available online 6 February 2016

Keywords:
Rasmussen
Systems theory
STAMP
Intent specifications

ABSTRACT

This paper describes three applications of Rasmussen's idea to systems engineering practice. The first is the application of the abstraction hierarchy to engineering specifications, particularly requirements specification. The second is the use of Rasmussen's ideas in safety modeling and analysis to create a new, more powerful type of accident causation model that extends traditional models to better handle human-operated, software-intensive, sociotechnical systems. Because this new model has a formal, mathematical foundation built on systems theory (as was Rasmussen's original model), new modeling and analysis tools become possible. The third application is to engineering hazard analysis. Engineers have traditionally either omitted human from consideration in system hazard analysis or have treated them rather superficially, for example, that they behave randomly. Applying Rasmussen's model of human error to a powerful new hazard analysis technique allows human behavior to be included in engineering hazard analysis.

1. Introduction

New ideas in engineering often appear in multiple places around the same time when the world is ready for and needs them. A systems approach to safety appeared at the same time in several different contexts: Rasmussen in human factors and safety, von Bertalanffy (1968) in biology, and System Safety in the American ballistic missile program. The primary factor driving these ideas was greatly increased complexity in the systems we were building. The old approaches simply did not scale.

Rasmussen was trained in electrical and control engineering and much of his work reflects that engineering training. I believe his greatest achievements are in integrating human factors and engineering and applying the resulting ideas to safety. However, his ideas have had the most impact on human factors and cognitive systems engineering and not on the engineering of the hardware and software beyond the interface design. My goal has been to apply the ideas beyond human factors in order to influence the way that engineers approach the entire engineering process for complex, safety-critical sociotechnical systems.

For the most part, system safety engineering (beyond the physical interface design) has either ignored the human operators or has treated human factors as an independent topic that is an appendage or parallel to the system engineering process. System engineering concentrates on designing hardware and software to prevent accidents. When the inevitable accidents result from the lack of consideration of humans in system design, the losses are simply blamed on the human operators. Most of the time, in fact, operator error can be directly traced to flaws in the technical design of the system. We will make only limited progress in system safety until we enlarge the scope of system engineering. Rasmussen provided clues about how to accomplish this goal.

My own attempts to extend Rasmussen's ideas to engineering practice have involved improving engineering specifications, particularly the requirements engineering process; creating a new, more powerful, model of accident causation that better explains the cause of accidents in human-operated, software-intensive, sociotechnical systems; and creating new hazard analysis techniques that integrate humans into the generation of causal scenarios. I describe each of these briefly in this paper, but first it is necessary to understand some basic ideas in systems theory, which is what underlies Rasmussen's (and my) approach.

2. An introduction to systems theory

Systems theory first arose in the 1940s and 1950s as a reaction to the changes in engineering that were starting to appear at that time. The most important change was in the complexity of the systems we were building. As complexity increased, traditional engineering approaches to system design and analysis became less and less effective.

E-mail address: leveson@mit.edu.

http://dx.doi.org/10.1016/j.apergo.2016.01.015
0003-6870/

Traditionally, science and engineering have coped with complexity in two ways: analytic reduction and statistical analysis. Analytic reduction goes back to the time of Descartes and is what most scientists and engineers are taught. The strategy is basically to handle complex systems by (1) dividing the system into distinct parts for analysis, (2) examining the parts separately, and later (3) combining the separate analysis results to provide results for the system as a whole. The physical aspects are divided into physical components, the functional aspects into functional components, and behavior is treated as distinct events over time.

The usefulness of analytic reduction rests on several assumptions: (1) Each component or subsystem operates independently, (2) the analysis results are not distorted when considering the components separately, (3) the components act the same when examined singly as when playing their part in the whole, and (4) the events are not subject to feedback loops and non-linear interactions.

Traditional approaches to safety engineering were created using this approach. Accidents are assumed to be caused by component failure so the analysis involves dividing the system into components and identifying chains of directly related physical or functional component failures that can lead to a loss. Failure events are assumed to be random with a constant failure rate (exponentially distributed) so that the probability of a loss can be derived. After the component failure scenarios are identified, engineers use fault tolerance or fail-safe design techniques to protect against the component failures identified in the accident scenarios. Humans and social factors were for the most part omitted from consideration.

This approach to safety made sense for the relatively simple, pure electro-mechanical systems of the past where system components could be effectively decoupled, allowing simple, direct interactions among components. System design errors could, for the most part, be identified and eliminated by testing and what remained after development were random hardware failures. Operational procedures could be completely specified and human error mostly involved skipping a step or performing a step incorrectly. Reliability and safety were, therefore, closely related in these relatively simple designs.

This situation is now changing. Software is becoming an integral part of most systems, and it allows enormously more complex systems to be constructed; humans are increasingly assuming supervisory roles over automation, which requires more cognitively complex human decision making; and accidents are more often resulting from unsafe interactions among components and not just individual or multiple component failures. While software design errors may exist that result in the software not implementing the stated requirements, the role of software in accidents and safety-related incidents is much more likely to result from inadequate software requirements. The software can be perfectly reliable (will do the same thing continually given the same inputs), but may still be unsafe.

The problems are similar for human operators. Assumptions about the role of human operators in safety have always been oversimplified. Most (many?) human factors experts now accept the fact that behavior is affected by the context in which it occurs and humans do not "fail" in a random fashion (see, for example, Dekker (2006), Flach et al. (1995), Norman (2002), Rasmussen (1997)).

The basic problem is complexity. Complexity has increased in current advanced engineering systems to the point where all the potential interactions among system components cannot be anticipated, identified, and guarded against in design and operations. *Component interaction accidents* (as opposed to *component failure accidents*) are occurring where no components have "failed" but a system design error resulted in losses caused by previously unidentified, unsafe component interactions and component requirements specification errors. Hazard analysis techniques based on reliability theory and assumptions that accidents are caused by component failures do not apply to component interaction accidents.

As a result of these changes, new types of accidents are occurring and, in particular, are resulting from new causal factors, such as mode confusion or requirements incompleteness flaws (often missing cases) where nothing "failed" but the problem was in the overall system design. System theory handles these types of design problems.

Systems theory was created as an alternative to analytic reduction (von Bertalanffy, 1968; Weiner, 1965; Ackoff, 1971; Checkland, 1981). It focuses on systems taken as a whole, not on the parts considered separately. Systems theory assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facets and relating the social to the technical aspects [Ramo 1973]. These system properties derive from the relationships among the parts of systems: how the parts interact and fit together [Ackoff 1971]. Thus the systems approach concentrates on the analysis and design of the whole as distinct from the components or parts.

Some properties in complex systems are *emergent*, that is, they arise from the interactions among the components. Safety is an emergent property. Looking only at one part of a complex system, it is not possible to tell whether an accident will occur. The property of "safety" only makes sense when all the interactions among the system components are considered together. The concept of emergence gives rise to the often quoted basic systems theory principle that in complex systems "the whole is greater than the sum of the parts."

Fig. 1 depicts a system or process made up of components (the shaded boxes). The components interact in both direct and indirect ways. Emergent system or process properties arise from these interactions.

In order to ensure that the system is safe, constraints on the component interactions must be enforced. This need for enforcing safety constraints implies that there is some kind of "controller" that enforces the safety constraints by handling individual component behavior (including failures) and component interactions (Fig. 2). Safety constraints are statements about what types of system level behavior or states are unacceptable, for example, power must never be on when the access door is open,

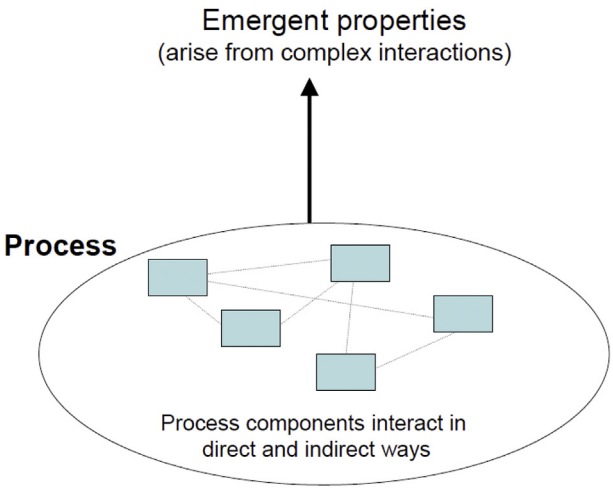

Fig. 1. Emergent properties arise from component interactions.