Association for Information Systems
AIS Electronic Library (AISeL)

ICIS 1999 Proceedings

International Conference on Information Systems (ICIS)

December 1999

Complexity in Embedded Intelligent Real Time Systems

Erman Coskun
Rensselaer Polytechnic Institute

Martha Grabowski
Rensselaer Polytechnic Institute and LeMoyne Colelge

COMPLEXITY IN EMBEDDED INTELLIGENT REAL TIME SYSTEMS

Erman Coskun
Rensselaer Polytechnic Institute
U.S.A.

Martha Grabowski
Rensselaer Polytechnic Institute and LeMoyne College
U.S.A.

Abstract

Embedded Intelligent Real Time Systems are proliferating in safety-critical large scale systems. Understanding and measuring the complexity in EIRTS can aid us in designing and building more reliable and effective EIRTS. In this paper, we propose a model of software complexity and describe a study of the impacts of EIRTS complexity on operator performance, software performance and system safety.

1. RESEARCH OBJECTIVES AND QUESTIONS

Embedded Intelligent Real Time Systems (EIRTS) are proliferating in many safety-critical large scale systems. ome examples include aviation, ship, or space shuttle control systems (Coenen, Smeaton and Bole 1989; Heudin 1991), air traffic control systems (Perry 1997), nuclear plant control systems (Wong and Kalam 1995), intelligent highway control systems (Dailey, Haselkorn and Lin 1993), flexible manufacturing systems (Ben-Arieh, Moodie and Chu 1988), patient monitoring system in intensive care units (Leveson and Turner 1993), and military and defense systems (Rouse, Geddes and Hammer 1990).

In all of these large scale systems, safety and reliability are important. During the last two decades, increasing attention has been paid to safety-critical large scale systems, particularly because of the enormous damage to human life, environment, property, and society associated with such events as the Chernobyl nuclear power plant accident in 1986, the Therac-25 Accidents between 1985-1987, and the *Exxon Valdez* oil spill in 1989.

EIRTS are often introduced in safety-critical large scale systems to improve the system's reliability and safety. However, EIRTS often also increase the system's complexity. Understanding and measuring the complexity in EIRTS can aid us in designing and building more reliable and effective EIRTS, particularly in safety-critical settings. In this paper, we propose a model of complexity in EIRTS and illustrate its use in evaluation of an operational EIRTS. We discuss the impact of complexity on EIRTS design and operation and conclude with expected contributions of this research.

2. THEORETICAL FOUNDATIONS

2.1 Embedded Intelligent Real Time Systems

An embedded software system is part of some larger system, which it controls and monitors (Highland 1994). Embedded intelligent systems reside in larger hosts, gather required data for reasoning from other components and from the environment, use their knowledge base to process and interpret data and reasoning, produce results, and send those results to other components or users.

Real time systems must satisfy explicit (bounded) response time constraints or risk severe consequences, including failure (Laplante 1992). Real-time computer systems automatically capture input data, as and when this data is available for capture, and deliver the processed information (Freedman and Lees 1977).

Embedded Intelligent Real Time Systems, thus, are systems that exhibit properties of each of these component systems. They are constructed from intelligent software components, are able to perform some functions with authority, and have the power to control and communicate in a constrained domain.

2.2 Software Complexity

Software complexity is a concept that has been defined in different ways by different disciplines. Halstead (1977) made the first systematic summarization of a branch of experimental and theoretical science dealing with the human preparation of computer programs. Mathematicians define software complexity based on dimensionality: the number of components and number of relationships among components in a software program (Kokol, Brest and Umer 1997; Zuse 1993). McCabe (1976) introduced the concept of "cyclomatic complexity," which was an application of graph theoretic complexity to computer software. Psychologists have defined software complexity in terms of its understandability by humans (Alford 1994; Banker, Davis and Slaughter 1998; Zuse 1991). Complexity poses problems for system users, developers, and maintenance personnel, as almost half of a software maintainer's time is spent trying to understand programs (Hirota et al. 1994). Cognitive psychologists have suggested that complexity, particularly for intelligent systems, should be addressed at the task, representation and implementation levels in order to address the different types of complexity in cognitive systems (Marr 1982). Economists often address complexity in terms of resource consumption, a measure of the resources that must be expended in developing, maintaining, or using a software product (Mildred 1996; Whitmire 1992). Computer and system scientists propose still other metrics of complexity, including the number of software errors (Mildred 1996), and the numbers of "operators," "operands," and/or the number of lines of code in a program (Gaffney 1982). In addition, Kolmogorov's (1965) complexity model has been applied to address the mathematical complexity of information.

Social scientists, particularly those of the "normal accidents" school, discuss the impact of technological complexity on socio-technical systems (Perrow 1984; Tenner 1996) and often describe complexity in terms of degrees of interaction and coupling. Interactions in a large scale system can be complex or linear, depending on whether events are unpredicted or unexpected. Coupling is the degree to which "reciprocal interdependence exists across many units and levels" and is a measure of the degree of slack and redundancy in the system (Perrow 1984).

These various authors have each approached software complexity in different ways. Large scale safety-critical systems are themselves complex entities. We suggest that understanding software complexity can aid us in designing reliable and thoughtful EIRTS and the large scale safety-critical systems in which they reside. It is this rationale that motivates our research.

3. RESEARCH METHODOLOGY

3.1 Proposed Model

Figure 1 illustrates the different domains that have contributed to understanding of software complexity and the approaches they have adopted. We propose that software complexity in EIRTS in safety-critical settings has important impacts on software performance, system safety, and operator performance. By measuring complexity in EIRTS, we can gain important insights as to the design of future safety-critical large scale systems, EIRTS software, and the human-technical systems that are important components of both.

In our research, we begin by addressing the impact of complexity on system safety through the use of social science, mathematical and system science approaches to complexity (Figure 2). We then propose to assess the impact of software complexity on software performance with the use of computer science, system science, economics, and mathematical metrics. Finally, we propose to examine the impact of software complexity on operator performance by examining complexity from an economic and psychological perspectives. We expect that, as in many large scale systems, the interactions between subsystems and components will provide at least as interesting results as the primary variables being studied.

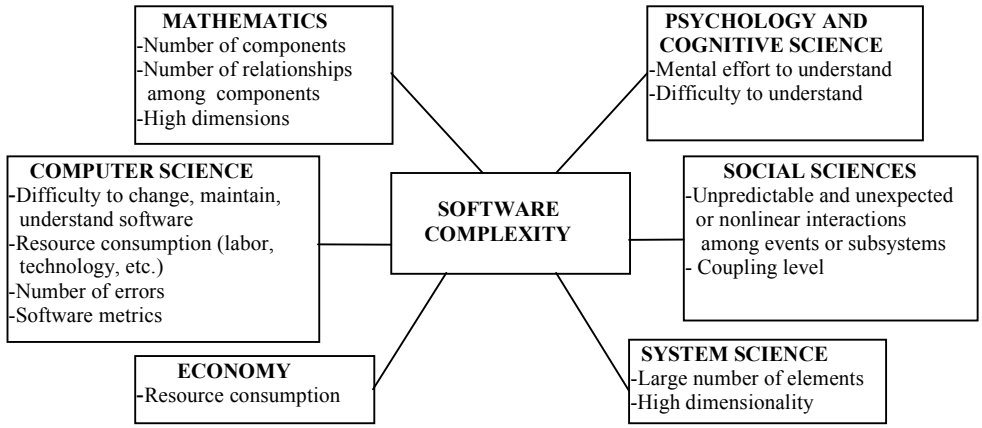

Figure 1. Representations of Software Complexity

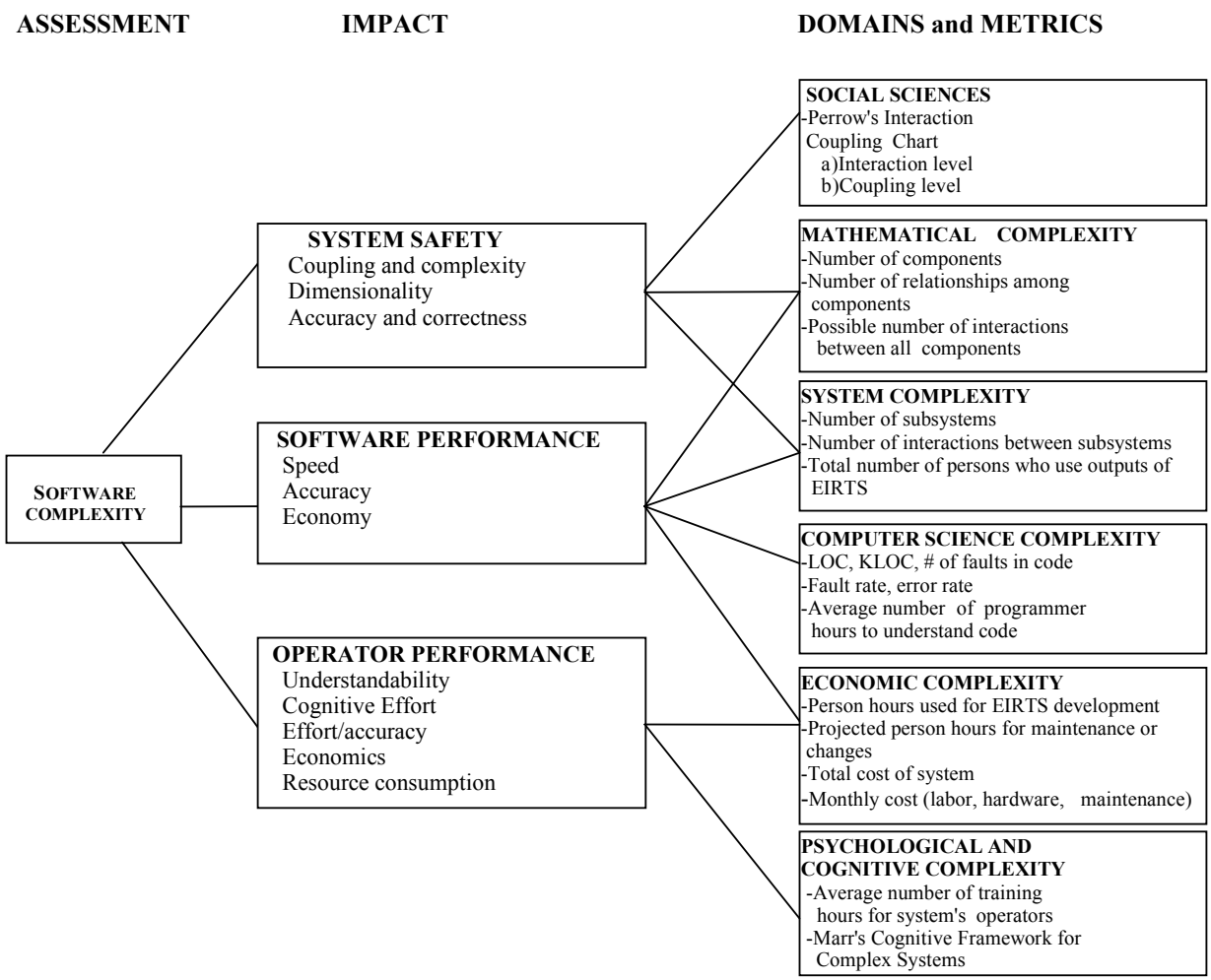

Figure 2. Proposed Model and Metrics For EIRTS Complexity

To illustrate use of the model, we will evaluate an operational EIRTS, the Navigation and Piloting Expert System (NPES), an embedded intelligent ship's piloting system developed for Lockheed Martin, the U.S. Defense Advanced Research Projects Agency, and Chevron Shipping Co. and installed aboard the *Chevron Colorado,* an operational oil tanker operating on the West Coast of the U.S. The NPES is a real-time knowledge based system that provides intelligent decision support to masters, mates and pilots navigating the restricted waters of San Francisco Bay. The NPES operates as an embedded module within the Lockheed Martin Smart Bridge™ Integrated Bridge System (IBS). It gathers real-time navigational, environmental, and sensor data, applies internally encoded knowledge to reason about the data, and provides real time recommendations and advice on all aspects of the transit to Chevron ship captains and San Francisco Bay Pilots.

3.2 Research Methodology

3.2.1 Subjects

There are three sets of subjects in this research:

(1) *Chevron Colorado* masters, mates, and San Francisco Bay pilots for operator performance related hypotheses,
(2) The NPES for the software performance related hypotheses, and
(3) The *Chevron Colorado* and other members of the San Francisco Bay Area traffic system (i.e., other vessels, vessel traffic controllers, ship's pilots, etc.) for safety related hypotheses.

3.2.2 Method

Hypotheses for each of the three primary research areas (operator performance, software performance, and system safety performance) were developed and are summarized in Figure 3. The operator performance evaluation will be a 3 x 2 design: three sets of subjects (masters, mates, and San Francisco Bar pilots) and two technology treatments (NPES and without NPES). The software performance evaluation will assess NPES performance with two different levels of software complexity. Finally, the system safety evaluation will evaluate safety in the San Francisco Bay vessel traffic system with the NPES and without the NPES.

After identifying the dimensions of software complexity and their potential impacts on system safety, software performance and operator performance, we will use domains in Figure 2 (along with our hypotheses, dependent variables and metrics in Figure 3) to measure the dimensions and impacts quantitatively. The appropriate statistical tests will be performed on collected data and the results of tests will be used to accept or reject our hypotheses.

4. SUMMARY

4.1 Contribution of Research

Multidisciplinary evaluations of EIRTS are seldom performed. The proposed model can offer insights to the dimensionality and impact of software complexity in EIRTS and on the large scale systems in which they are deployed. Such understanding can significantly increase our knowledge of how best to design, operate and evaluate EIRTS in safety-critical settings.

4.2 Current Status of Research

Currently, the literature review is completed and the proposed model to measure complexity is structured. The evaluation metrics have been determined. After finalizing of hypotheses, data collection is the next step in this research in progress.

HYPOTHESES	DEPENDENT VARIABLE	VARIABLE OPERATIONALIZATION
SYSTEM SAFETY HYPOTHESES		
1) The introduction of EIRTS in a large scale system will increase system complexity.	System complexity level	1) # of components **2) # of interactions among components** 3) # of people who benefit from system outputs **4) # of communications among crew members** 5) # of processes on raw data **6) Average time between data input and data output with and without EIRTS**
2) Increased system complexity can be correlated with increased level of system safety	System safety	1)# of accidents, incidents and unusual events **2) Crew confidence level with and without EIRTS**
3) Introduction of EIRTS will increase system levels of control in a large scale system	Levels of system control	1) Increased number of interactions among components of system **2) Average # of information coming to EIRTS under different scenarios (emergency vs. normal)**
4) Increased levels of system safety can be associated with systems that are loosely coupled and tightly coupled	.Levels of system control via changes on interaction and coupling levels with installation of EIRTS	1) Application of Perrow's chart and definitions of coupling and interaction to the marine transportation without EIRTS (Perrow already located Marine Transportation in the chart) and with EIRTS.
SOFTWARE PERFORMANCE HYPOTHESES		
5) Increased software complexity will result in poorer software performance	.Response time .Accuracy .Reliability .Response time/ accuracy .Use of system resources	1)Time to process a data with and without EIRTS 2) **# of timely responses** 3) Usability % of EIRTS advise
6) Design method and language may contribute to complexity of EIRTS and has effect on software performance	.Response time .Reliability .Efficiency .Effectiveness	1) Discussion of different design methods and languages **2) Number of faults, errors** 3) OO design advantages
7) Intelligent systems performance will be closely related to user performance	.User performance	1) Accuracy and correctness of decisions **2) Number of alternatives before decision making** 3) User confidence and effectiveness
8) EIRTS will increase the overall performance efficiency and effectiveness of real time large scale system	.Overall system performance	1) Speed of decision making **2) Accuracy of decisions** 3) Number of dangerous situations before and after EIRTS
OPERATOR PERFORMANCE HYPOTHESES		
9) Increased software complexity has a positive impact on operator performance	.Operator performance	1) Operator error rates **2) Task performance and task completion time** 3) Operator confidence and satisfaction **4) Required training hours** 5) Understandability of user interface
10) Economically complex software will increase operator performance	.Operator performance .Difficulty of tasks	1) NPES vs. SPES cost (Labor, # of people, time, yearly maintenance cost) **2) NPES vs. SPES operator performances (speed, quality and efficiency of decisions by operators)**
11) User understandability of software will be correlated to representation and implementation complexities	.Operator performance with user interface complexity	1) Marr's theory User interface complexity metrics **2) Hardware complexity**
12) Computer science complexity will increase operator performance	.Operator performance	1)Understandability of functions and commands of program, **2) Automation level of tasks by EIRTS (% of all tasks being supported by EIRTS)**

Figure 3. Hypotheses, Dependent Variables, and Operationalizations (Metrics) of Empirical Study

5. REFERENCES

Alford, M. "Attacking Requirements Complexity Using a Separation of Concerns," in *Proceedings of the First International Conference on Requirements Engineering*. 1994, pp. 2-5.

Banker, R. D.; Davis, G. B.; and Slaughter, S. A. " Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study," *Management Science* (44:4), April 1998, pp. 433-450.

Ben-Arieh, D. H.; Moodie, D. L;. and Chu, C. D. "Control Methodology for FMS," *IEEE Journal of Robotics and Automation.* (4:1), 1988, pp. 53-59.

Coenen, F. P.; Smeaton, G. P.; and Bole, A. G. "Knowledge-based Collision Avoidance," *Journal of Navigation (UK)* (41:1), 1989, pp. 107-116.

Dailey, D. J.; Haselkorn, M. P.; and Lin, P. "Traffic Information and Management in a Geographically Distributed Computing Environment," in *Proceedings of the Third Pacific Rim International Conference on Applications of Advanced Technologies In Transportation Engineering,* July 1993, pp. 159-165.

Freedman, A. L., and Lees, R. A. *Real-time Computer Systems*, New York: Crane Russak, 1977.

Gaffney, J. E. "Performance Evaluation Review," in *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (11:2), 1982, p. 31.

Halstead, M. H. *Elements of Software Science: Operating and Programming Systems Series*, New York: Elsevier Computer Science Library, 1977.

Heudin, J. C. "KOS: A Knowledge-based Operating System For Real-Time Onboard Applications of Artificial Intelligence," in *Proceeding of the Euromicro '91 Conference*, Los Alamitos, CA: IEEE Computer Society Press, 1991, pp. 138-144.

Highland, F. "Embedded AI," *IEEE Expert: Intelligent Systems and Their Applications* (2:3), 1994, pp. 18-20.

Hirota, T.; Tokhi, M.; Overstreet, C. M.; and Hashimoto, M. "An Approach to Predict Software Maintenance Cost Based on Ripple Complexity.," *Asia Pacific Software Engineering Confonference*, 1994, pp. 439-444.

Kokol, P.; Brest, J.; and Umer, V. "Computer Aided Analysis of Software Complexity," 1997 (retrieved July 20, 1998: http://mario.uni-mb.si/~metrics/articles/ecast97/ECAST97.html).

Kolmogorov, A. N. "Three Approaches to the Quantitative Definition of Information," *Problems of Information Theory* (1), 1965.

Laplante, P. *Real-time Systems Design and Analysis: An Engineer's Handbook*, New York: Institute of Electrical and Electronics Engineers, 1992.

Leveson, N. G., and Turner, C. S. "An Investigation of the Therac-25 Accidents," *IEEE Computer* July 1993.

Marr, D. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, San Francisco: W. H. Freeman and Company, 1982.

McCabe, T. J. "A Complexity Measure," *IEEE Transactions on Software Engineering* (SE-2:4), December 1976, pp. 308-320

Mildred L. G. "Practical Software Engineering Complexity—Software Metrics," 1996 (retrieved July 18, 1998: http://www.cpsc.ucalgary.ca/~mildred/451/Complexity.html).

Perrow, C. *Normal Accidents: Living With High-Risk Technologies*, New York: Basic Books, 1984.

Perry, T. S. "In Search of the Future of Air Traffic Control," *IEEE Spectrum*, August 1997.

Rouse, W. B.; Geddes, N. D.; and Hammer, J. M. "Computer-aided Fighter Pilots," *IEEE Spectrum* (27:3), 1990, pp. 38-41.

Tenner, E. *Why Things Bite Back: Technology and the Revenge of Unintended Consequences*, New York: Alfred A. Knopf, Inc., 1996.

Whitmire, S. "Measuring the Complexity of Object-oriented Software," in *Proceedings of the Third International Conference on Applications of Software Measurement*, 1992, pp. 2.207-2.242.

Wong, S. K., and Kalam, A. "Development of A Power Protection System Using An Agent Based Architecture," in *Proceedings of the First International Conference on Energy Management and Power Delivery*, 1995, pp. 433-438.

Zuse, H. *Software Complexity Measures and Methods*, Berlin: Walter de Gruyter Publishing, 1991.

Zuse, H. "Criteria for Program Comprehension Derived from Software Complexity Metrics," *Proceedings of the Second IEEE Workshop on Program Comprehension*, 1993, pp. 8-16.