2002

Assessment of Intelligence Complexity in Embedded Intelligent Real Time Systems

Erman Coskun
Le Moyne College, coskune@mail.lemoyne.edu

Martha Grabowski
Le Moyne College, grabowski@mail.lemoyne.edu

ASSESSMENT OF INTELLIGENCE COMPLEXITY IN EMBEDDED INTELLIGENT REAL-TIME SYSTEMS

Erman Coskun
Le Moyne College Business Department
Syracuse, NY 13214 USA
Phone: 315 445 4790 Fax: 315 445 4540
E-mail:coskune@mail.lemoyne.edu

Martha Grabowski
Le Moyne College Business Department
Syracuse, NY 13214 USA
&
Rensselaer Polytechnic Institute
Decision Sciences and Engineering Systems Department
Troy, NY 12180
E-mail:grabowsk@mail.lemoyne.edu

ABSTRACT

Intelligent systems and their applications are proliferating. Embedded Intelligent Real-Time Systems (EIRTS) are one type of intelligent system. Defining and measuring the complexity of this kind of system may help with better design, development, maintenance, and performance of EIRTS. In this paper, we propose a set of evaluation criteria to measure the complexity of Embedded Intelligent Real-Time Systems (EIRTS). We show an operationalization of the criteria with a sample EIRTS.

1. INTRODUCTION AND RESEARCH QUESTIONS

Many next generation real-time systems are expected to be large, complex, distributed, intelligent, and able to operate in increasingly uncertain environments [Grosv & Davis, 1994; Stoyen, Marlowe, Younis, and Petrov, 1999]. These real-time systems must be intelligent and flexible enough to react and respond quickly to changing and unexpected system conditions, must evolve over time as requirements change, and must keep development, testing, and verification costs low. Embedded Intelligent Real-Time Systems (EIRTS) are one example of such systems. EIRTS process data quickly, reason about the processed data, and use the results to provide decision support, system monitoring, or system management capabilities. EIRTS have been deployed in a wide range of systems to solve complex problems, to speed up data processing, and to enhance system reliability and usability. One of these deployment areas is the realm of safety-critical large-scale systems. Examples of these systems include intelligent highway systems, nuclear power plants, vessel traffic control systems, distributed manufacturing systems, battle management, and patient monitoring systems. Successful measurement and interpretation of complexity in intelligent systems can assist with the development of more reliable and safer systems, significantly reduced cost of maintenance, and better design and performance for both the EIRTS and for the large-scale safety-critical systems within which they are embedded. In this study, we focus on assessing the complexity of EIRTS deployed in safety-critical large-scale systems. We start with a literature review as background to this research. After proposing a set of evaluation criteria, we operationalize the set with a sample EIRTS. The last section includes conclusions and contributions of this study and a conference presentation plan.

2. THEORETICAL FOUNDATIONS

2.1 Software Complexity

Software complexity is a concept that has been defined in different ways by different disciplines. Software complexity measures attempt to objectively associate a number with a program, based on the degree of presence or absence of certain characteristics of software [Kokol, Brest, and Umer, 1996]. These characteristics may change with the point of view. For example, for software engineers, the complexity might mean the number of errors in the code, or the required development cost and time, while for a cognitive scientist or for a human interface expert, the complexity might have to do with difficulty in understanding the software. In this study, we view software complexity as the complexity introduced into systems with embedded intelligence. This complexity therefore includes errors in code, difficulty with understanding the software, and complexities related to the system's intelligence.

Intelligent systems are the focus for our interest in software complexity. Intelligent systems may be designed for different purposes and functionality. They may be designed to function as a control unit of a large system, as a decision support system to help a decision-maker in a system, as a monitor of components in a system, or as an intelligent part of a system.

Since intelligent systems are relatively recent developments, much intelligent system research has focused on system design and development issues. Recently, however, these systems have reached maturity levels where research related to their evaluation is possible [Grabowski and Sanborn, 2001]. Thus, intelligent systems complexity research is also a relatively new area of research.

2.2 Previous Work

Marr [1982] provides an early framework for intelligent systems evaluation. He suggests that intelligent systems should be evaluated at the ***task level***, assessing what system does and why it does it; at the ***representation level***, focusing on the logical organization of coding structure used for knowledge representation; and at the ***implementation level***, examining the system's algorithms and representations. In contrast, Reich [1995] identifies two types of knowledge in intelligent systems that contribute to complexity -functional and structural knowledge. Functional knowledge cannot be measured directly, but only indirectly by measuring the behavior of a system that has knowledge, while structural knowledge is a static entity that includes facts, rules and models that represent real world phenomena [Reich, 1995]. Strainieri and Zeleznikow [1999] follow a similar approach and suggest qualitative and quantitative metrics of knowledge complexity: readability to the experts is proposed as a qualitative metric for structural knowledge and the number of rules and rule correctness are proposed as quantitative metrics. For functional knowledge, Strainieri and Zeleznikow propose problem-solving behavior as a qualitative metric and comparisons with expert decisions, assessments of user satisfaction, and user acceptance as quantitative metrics. Both functional and structural knowledge contribute to the complexity of intelligent systems, and are important elements to consider in evaluating the complexity of an intelligent system.

Chen and Suen [1994] focus on the complexity of rule-based expert systems and propose metrics such as the number of rules, average depth of search space, and the number of matching patterns contained in set of connected rules as complexity metrics. Barr [1999] proposed a graph representation of the complexity of rule-based expert systems, claiming that other graph-based metrics such as McCabe's cyclomatic complexity [McCabe, 1976] cannot adequately determine the number of execution paths in a rule base. Finally, Sharma and Conrath [1996] proposed a socio-technical model for evaluating expert systems, using measures of user satisfaction, effectiveness, value and utilization. Thus, a number of authors have identified qualitative and quantitative measures of complexity for intelligent systems by using different perspectives.

2.3 Intelligence in EIRTS

Based on this literature review, we consider three intelligence in EIRTS from 3 perspectives. First, the EIRTS reasoning can be considered intelligent. Second, the decision support provided by the EIRTS can be considered intelligent. Finally, the user interface and the human-computer interaction provided by the EIRTS can be considered intelligent. These three dimensions provide both functional (decision support and human-computer interaction) and structural measure of complexity, utilizing quantitative and qualitative measures.

EIRTS reasoning intelligence is related to data processing, result production and interpretation, and use of the processed data to help users or decision-makers. EIRTS take raw data from system components and/or the outside environment, process that data, produce results from data by applying reasoning algorithms and strategies, and make necessary decisions (in automated control systems) or display results to users (in decision support systems). The complexity of intelligent reasoning can be measured by code analysis and functions and structures in the software can be used to assess it.

A second part of EIRTS intelligence is related to the usability of EIRTS output. After an EIRTS produces output, a user must understand its advisories, the logic behind the advice, and then easily transfer this information to cognitive thinking and decision-making. Usability directly impacts the decision quality and performance of users. The complexity associated with the intelligent human-computer interface can be measured using metrics from the decision support systems, computer aided decision-making, cognitive science, psychology, and human-computer interaction literature.

EIRTS also provide information and support to users and/or decision makers. In order to make good decisions, user should be well informed about the situation, alternative decisions should be determined and evaluated, and the results of each alternative must be presented. EIRTS help users with information, alternative determination, alternative testing, and result prediction stages. The decision support provided by EIRTS can also therefore be considered intelligent. The complexity of decision support intelligence can be measured by using metrics such as the timeliness of software advice, the degree of users' understandings of software output, the users' ease of interpreting the advice provided by the system, users' perceptions of the quality of their decisions, the efficiency and effectiveness of users' decision-making processes with the software, and the users' perceptions of support provided for different types of decisions.

The roots of EIRTS intelligence - its reasoning, human-computer interface, and decision support capabilities- suggest that data and input for intelligence complexity measurements must come from three sources: user feedback, code analysis, and user and system performance assessments. In this study, we conduct experiments using these 3 data sources to assess complexity levels in Embedded Intelligent Real-time Systems.

3. THEORETICAL MODEL

Following Marr [1982], we consider the intelligence complexity of EIRTS at the system's task, representation, and implementation levels. The EIRTS tasks are the activities and functions that it performs; assessing task complexity, therefore, involves investigation of the EIRTS' functionality and how well it performs those tasks. EIRTS generally perform three types of tasks in safety-critical large-scale systems: data processing and interpretation, decision support, and system monitoring. Most EIRTS applications involve *data processing and interpretation*: gathering data and reasoning about that data using system knowledge. For instance, intelligent highway control systems measure traffic data on the highway, compare that data, and reason about the processed data to manage traffic on the highway [Dailey, Haselkorn, and Lin, 1993]. EIRTS also perform reasoning to support *decision-making*. To accomplish this, EIRTS reason about collected data and provide the results of that reasoning and information to decision-makers.

EIRTS Level	Description	Types of Metrics	Data Source
Task	• What the EIRTS does and how/why it does it	• **Functional Performance** ➢ Data Processing ➢ Decision Support • **Decision quality**	• **CODE ANALYSIS** • **USER OPINION**
Representation	• Logical organization of coding structures used for knowledge representation	• **Algorithmic complexity** • # of If-Else Statements • Cyclomatic complexity • **Number of include statements** • **Coupling between objects**	• **CODE ANALYSIS**
Implementation	• Adequacy and architecture of built system • Performance as built	• **Structural complexity** ➢ Number of functions, classes, methods • **Response time** • **Reliability of output** • **Accuracy of output** • **User understandability** • **User perception of support level** -Support for situation Monitoring -Support for Threat Determination -Support for threat Avoidance -Support for Maneuvering • **Difficulty to use** • **Cognitive skill requirement**	• **CODE ANALYSIS** • **USER and SYSTEM PERFORMANCE**

Table 1. EIRTS Intelligence Complexity Evaluation Metrics

One example of this type of EIRTS application is an intelligent shipboard piloting system, which gathers data, reasons about that data, and provides advice and recommendations for ship control to the ship's master and pilot [Grabowski & Sanborn, 2001]. Assessing the task complexity of an EIRTS, therefore, requires the use of functionality metrics, such as the number of methods and number of functions supported, as well as process and outcome metrics such as user satisfaction and user confidence with the EIRTS' data processing, and decision support (Table 1). To gather this data, both code analysis and user opinion are required.

Representation-level complexity evaluations focus on the complexity of the algorithms, knowledge structures, control mechanisms, and feedback systems in the EIRTS. The algorithmic complexity of intelligent systems has been much studied, with a variety of metrics devised (i.e., McCabe's cyclomatic complexity [McCabe, 1976]). For representation-level metrics, the studies mentioned previously propose metrics related to the size, depth, type, and content of knowledge. Assessing the complexity of an EIRTS' representation, therefore, requires use of metrics that focus on the algorithms, knowledge structures, control mechanisms and feedback systems in the EIRTS. This is primarily determined through the use of code analysis.

Implementation-level complexity evaluations focus on the adequacy and architecture of the built system, and its performance as built. Evaluating the implementation complexity of an EIRTS requires the use of metrics that evaluate the structural complexity of the EIRTS (eg., numbers of functions, classes, methods, interfaces) as well as the performance of the EIRTS (eg., time and speed of data processing, decision support, task support level, difficulty, cognitive skill requirements, and

understandability). Implementation level complexity is primarily determined by code analysis, as well as assessments of user and system performance. Assessing the complexity of an EIRTS, therefore, requires the use of several types of metrics, at the task, representation, and implementation levels, as shown in Table 1.

4. RESEARCH VEHICLE

As a way of operationalizing the proposed Table 1 evaluation criteria, we use as a research vehicle the Navigation and Piloting Expert System (NPES), an operational EIRTS developed by Rensselaer Polytechnic Institute as part of the Lockheed Martin SmartBridge™ initiative [Spotts and Castellano, 1997]. The NPES is a real-time intelligent ship's piloting system that provides intelligent decision support to masters, mates on watch, and ship's pilots navigating the restricted waters of San Francisco Bay. The NPES was embedded within a real-time ship control system known as the SmartBridge™, which provides navigational and piloting support to ship's bridge watch teams.

Two versions of the NPES were developed, NPES-1 and NPES-2. NPES-1 was the original implementation of the EIRTS that was deployed aboard a tankship in the U.S. West Coast oil trade, the *Chevron Colorado.* After NPES-1 was deployed, additional design, functionality, reasoning, and structural changes were made, resulting in a subsequent version of the EIRTS, named NPES-2.

For instance, the NPES-2 reasoning algorithm was changed from that used in NPES-1 by adding target clustering technology, and by improving multiple-target collision avoidance algorithm. Also some new functionality was added to provide more detailed information such as ownship position and maneuvering limitations. These changes resulted in changes in file numbers, sizes of files, numbers of classes, numbers of attributes, and numbers of public, private, and protected methods between the two versions. The data processing/reasoning/functionality changes between NPES-1 and NPES-2 mostly focused on the NPES maneuver generating algorithm and data processing areas. NPES-2 has better control of the time for maneuver generation. It automatically stops the recommendation generation if a predefined time is passed, so that NPES is reasoning about "current" targets and problems.

Another difference between the two versions can be seen in the user interface and screen design. On the user interface side, the changes in NPES-2 were mostly done to provide better information to the users. The first difference between the two versions' user interface was the chart used. NPES-1 uses a raster image digital chart, which looks like the charts used in daily life by pilots and navigators. NPES-2 uses a fully vectorized electronic chart display information, which gives a less real-life chart image. There were also changes in message display style to provide better and more noticeable information to the users. Blinking and red colored warnings, alerts, and alarms were used in NPES-2, while NPES-1 used regular characters and black color for displaying these information. In terms of information content, both NPES-1 and NPES-2 display the same one sentence warning for alerts/alarms. However, in NPES-2, if the user wants to learn more details about that situation, additional information is available by clicking on the warning, alert/alarm sentence. NPES-2 also provides additional operator information on the bottom of the NPES screen about ownship's current location, speed, bearing, and CPA information to the users. NPES-1 does not have this feature. The main screen designs also have some differences in NPES-1 and NPES-2 . In NPES-1, there are two main menu items located on the SmartBridge menu. These are "NPES" and "ADVISORIES". The NPES choice opens the NPES recommendations screen, and shows alerts/alarms, recommendations. The ADVISORIES choice opens three windows and shows "required tasks", "environmental conditions", and "local/pilot knowledge" advisories simultaneously. The user may select and view either NPES or ADVISORIES screen. However, if ADVISORIES screen is open, users cannot see any alert/alarm warning unless they switch to the NPES screen. In NPES-2, alerts/alarms information is displayed in a fixed window and always available. In NPES screen, there are 4 menu items. They are "NPES Alert Info" which shows warnings, alerts/alarms, recommendations, "Local/Pilot Knowledge", "Required Tasks", and "Environmental Info" which show advisories. The user may select any of them, and can switch between them depending on situation and information need.

Another display difference between NPES-1 and NPES-2 is the display of ownship and target vessels. NPES-1 shows ownship and target vessels without any direction arrow. This display may confuse users, since they cannot be sure about the direction of ownship and targets. In NPES-2, arrows are added to ownship and target displays. The tip of arrow shows the direction and heading of vessels. The last difference between NPES-1 and NPES-2 is the NPES On/Off switch. In NPES-2, if the user does not want to see NPES information, they may click on this button and turn NPES off. This button is not available in NPES-1.

5. DATA COLLECTION AND MEASUREMENTS

Three types of data at the task, representation, and implementation levels were obtained based on the Table 1 evaluation criteria. Three types of experiments were run to test the model: User feedback and code analysis to determine task level complexity; code analysis for representation level complexity; and code analysis and user and system performance measures for implementation level complexity.

For user performance 3 experienced navigation users were run through a total of 8 scenarios comparing the two NPES versions. For code analysis, appropriate metric values for code complexity were collected for both NPES versions. For system performance measures, 8 different scenarios were run with both systems and performance-related data collected for the system and for the users.

6. RESULTS

Based on the Table 1 evaluation criteria, we calculated intelligence complexity metric values for both NPES versions and present their summary in Table 2. As can be seen in that table, for most of the metrics, NPES-2 is more complex on all three levels of complexity. For the task level complexity, metrics both coming from user evaluations and from code analysis were used. The metrics show that users' confidence is significantly higher with NPES-1, while functional and decision support differences are not significant between NPES-1 and NPES-2. The users believe that the quality of their decision will be higher with using NPES-1, and they report that they are more satisfied with NPES-1 functionality and decision support. This significant preference of NPES-1 by users is not because of better intelligence of it, but because of other complexities such as usability and decision support/explanation complexities.

The results for representation level complexity are gathered mostly from code analysis. They suggest that the architecture and reasoning structure of NPES-2 is more complex than NPES-1, but not significantly. This complexity is mainly the result of changes and additions between the two versions, and the results are parallel to users' and developers' feedback. However, although NPES-2 values for all selected representation metrics are greater than NPES-1 metric values, the differences between metric values are not significant for any of the metrics. Thus, although the NPES-2 representation is slightly more complex than NPES-1, the differences are not statistically significant.

Implementation level complexity calculations were derived from code analysis and system performance measurements. These calculations show that for some metrics, the NPES-2 implementation is more complex than NPES-1, primarily because of additional design, functionality, data processing, reasoning, and structural changes on NPES-1. This result also supports the coders' and implementers' opinions with respect to NPES-2 functionality, code structure design, and algorithms. Average response time and maximum response time values are significantly higher for NPES-2. This is because of more required processing time, more complex algorithms, and more complex reasoning. However, the user acceptance rate of NPES recommendations is higher for NPES-1. This is contrary to expectations, and it shows that since implementation complexity is higher for NPES-2, these users prefer less complex implementations. The results from user interface complexity also supports these findings.

These findings are consistent with the post-experiment interviews, where subjects reported that since NPES-1 user interface and especially the NPES-1 chart provided more raw data, they felt they were making decisions with more information with NPES-1. The results also show that NPES-1

recommendations were accepted by subjects at a significantly higher rate than NPES-2 (p-value **0.0416**), indicating that subject decisions were supported better with NPES-1. The subjects' ratings also show that NPES-1 support for situation monitoring, threat avoidance, and maneuvering tasks -- the key elements of navigation decision support-- are significantly better (p-values=**0.0026, 0.008, 0.046** respectively), also indicating better decision support for users.

Level	Metric	Metric Values NPES-1[1]	Metric Values NPES-2[1]	t-value[2]	p-value[2]
Task Level Complexity	• **Functional Performance**				
	➢ Data Processing				
	Size of Knowledge-base classes	1487	1685	.384	0.705
	Number of changes made in intelligence classes				
	Number of functions in intelligence classes	16.61	15.3	.354	0.727
	➢ Decision Support				
	Number of Advisories	110	114	-1.095	0.353
Representation Level Complexity	• **Algorithmic complexity**				
	Number of IF statements	56.5	74	-0.759	0.457
	Number of ELSE statements	15.8	18.44	-0.584	0.566
	Cyclomatic complexity	121.33	183.6	-0.541	0.595
	• **Number of include statements**	10.5	11.81	-0.965	0.345
	• **Coupling between objects**	22.07	23.27	-0.218	0.830
Implementation Level Complexity	• **Structural complexity**				
	➢ Number of methods,	16.6	15.45	.736	0.342
	➢ Number of attributes	8.77	8.10	.746	0.380
	• **Average response time**	4.27	8.85	**-5.96**	**0.001**
	• **Maximum response time**	8.85	18.85	**-2.11**	**0.028**
	• **Acceptance rate for recommendations**	4.95	3.25	**0.815**	**0.041**
		5.13	3.90	**2.789**	**0.021**
	• **Accuracy of output**	5.67	4.67	1.061	0.174
	• **User understandability**				
	• **User perception of support level**	6.67	3	**5.5**	**0.0026**
	-Support for Situation Monitoring	5.67	4	0.945	0.199
	-Support for Threat Determination	6.33	3.33	**4.025**	**0.008**
	-Support for Threat Avoidance	6.0	3.66	**2.21**	**0.046**
	-Support for Maneuvering	4.0	3.66	0.5	0.322
	• **Difficulty to use**	6.33	6.33	0	0.5
	• **Cognitive skill requirement**				

[1]Likert Scale questions designed as 1 less desirable 7 most desirable
[2]Bold number show statistically significant results

Table 2. Sample Results

7. CONTRIBUTION, CONCLUSIONS AND CONFERENCE PRESENTATION

This paper proposes a set of evaluation criteria to measure the intelligence complexity of EIRTS. Our comparison of two versions of an operational EIRTS highlights the differences in intelligence complexity at the user, code, and system performance levels. The results help us to understand the complexity of EIRTS intelligence better, which can be used for different purposes such as increasing system performance, decreasing complexity, and providing better decision support and usability to operators.

Also, all 3 users reported that they prefer to work with NPES-1 because it was better supporting their decisions, less complex, and easy to use. Since our preliminary results show that NPES-2 has a higher intelligence complexity, further analysis are required (being conducted) in order

to explain the relationship between intelligence complexity and user preferences as well as impacts of intelligence complexity on user performances.

We strongly believe that our results can be easily used as a framework and could be applied to all kind of intelligent systems such as decision support systems and expert systems. Of course each individual system should be studied based on its characteristics and constraints, and this would affect metric selections and methodology slightly but the framework from this study can be easily adjusted. Practitioners interested in studying their systems can use the developed model, and follow the presented methodology to determine details of their study.

Currently, system performance measurements and further analysis are underway. We are also applying this framework to another intelligent system in order to determine intelligence complexity. The details of evaluation criteria, metrics, and results, along with system performance measurements, will be presented during our conference presentation.

REFERENCES

Barr, V. (1999) Applications of Rule-base Coverage Measures to Expert System Evaluation K*nowledge-Based Systems* 12 pp.27-35

Chen, Z. and Suen, C.Y. (1994) Complexity Metrics for Rule-Based Expert Systems *International Conference on Software Maintenance* 94 pp.382-391

Dailey, D.J. Haselkorn, M.P. and Lin, P. (1993) Traffic Information and Management In A Geographically Distributed Computing Environment. Proceedings of the Pacific Rim Trans Tech Conference on Object-Oriented Programming Systems, Languages and Applications Jul 25-28 1993 1993 Seattle, WA, USA, Published by ASCE New York NY USA pp: 159-165 ASBN : 0-87262-916-3

Grabowski, M.R., & Sanborn, S.D. "Evaluation of Embedded Intelligent Real-Time Systems," *Decision Sciences*, 32:1, Winter 2001, pp. 95-123.

Grosv, B. and Davis, R. (1994). A Report to ARPA on Twenty-First Century Intelligent Systems. *AI Magazine*. 15(3). pp. 10-20

Kokol, P., Brest, J., and Umer, V., Software Complexity – An Alternative View, *ACM SIGPLAN notices*, ACM Press, Volume 31, Num. 2 (1996) pp. 35-41

Leveson (1995) Safeware: System Safety and Computers, Addison-Wesley, 1995

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

McCabe, T.J. (1976) A Complexity Measure, *IEEE Transactions On Software Engineering*, Vol. SE-2, No.4 (December 1976) pp.308-320.

Reich, Y. (1995) Measuring The Value of Knowledge *International Journal of Human-Computer Studies* vol.42 pp.3-30

Sharma, R.S. and Conrath, D.W. (1996) Some Soft Measures for Performance Analysis: The "Core" Dimensions of Expert System Quality *Microelectronic Reliability* vol.36 no.6, pp.775-796

Spotts, T.E.; Castellano, C. (1997) SmartBridge - Navigation safety *Sea Technology* v 38 n 11 Nov 1997 Compass Publ Inc. pp. 58-62

Stoyen, A.D.; Marlowe, Th.J.; Younis, M.F.; Petrov, P.V. (1999) Development environment for complex distributed real-time applications, *IEEE Transactions on Software Engineering*, Volume 25, Issue 1, January - February 1999, pp. 50-74

Stranieri, A. and Zeleznikow, J. (1999) The Evaluation of Legal Knowledge-based Systems *Proceedings of the seventh international conference on artificial intelligence and law* June 14 - 17, 1999, Oslo Norway pp.18-24 ACM

Wong, S.K. and Kalam, A. (1995). Development of A Power Protection System Using An Agent Based Architecture. *Proceedings of International Conference on Energy Management and* Power Delivery. pp. 1, 433-438.