New Paradigm of Common Cause Human Behavior Error Domain in Human-Software Interaction

Peom Park†, Kwan Suk Lee‡

†Department of Industrial Engineering, Incheon University,

‡Department of Industrial Engineering, HongIk University

ABSTRACT

This study is to develop a cognitive paradigm including a new model of common cause human behavior error domain and to analyze their causal factors and their properties of common cause human error characteristics in software engineering.

A laboratory study was performed to analyze the common causes of human behavior domain error in software development and to identify software design factors contributing to the common cause effects in common cause failure redundancy.

The results and analytical paradigm developed in this research can be applied to reliability improvement and cost reduction in software development for many applications. Results are also expected to provide training guidelines for software engineers and for more effective design of ultra-high reliabile software packages.

I. INTRODUCTION

This study introduces the analysis and a new domain paradigm of common-cause human behavior error in human-software interaction. This study is concerned with common-cause human behavior domain errors during software system development. This includes the contents, conditions, and their characteristics in human software interaction. It also concerns interactions between the human, who is presumed responsible for overseeing the software system, usage of the software system and software development. Since the software components are not independent of each other in regard to failure behavior, software redundancy does not improve reliability except in multi-version software development. Multi-version software system development is often requested to improve of reliability, especially in ultra-high reliability

systems such as nuclear power control, air traffic control, space shuttle missions, and war games. The major common-cause human behavior domain errors found in this research can contribute strongly to internal common-cause failure effects in a multi-version software development project.

The common-cause error model includes three analytical reasoning categories and a common-cause function established in terms of human-software information processing systems, human error mechanisms, and cognitive control domains. It is used to characterize the human factors mechanisms behind typical categories of errors considered as occurrences of human-software task mismatches.

Recently, Deborah Mitta(Mitta, 1991) presented a methodology for quantifying expert system usability which is considered from a designer's prospective. A linear multivariate function for measuring usability is described and procedures for selecting function variables are provided. The usefulness of the usability function as a design tool is investigated. The six variables for expert useability are: user confidence, the user's perception of difficulty, correctness of solution, the number of responses required of users, inability of expert system to provide a solution, rate of help requests. Jens Rasmussen(Rasmussen, 1987) classified cognitive control domains: skill, rule, and knowledge based behavior. He also described psychological mechanisms in the area of human-task mismatches. Modeling and predicting human error was studied by David D. Woods(Woods, 1990). This research included a limited rationality approach and some directions in error modeling. James Reason(Reason, 1987) studied a general framework for locating the principal limitations and biases giving rise to the more predictable varieties of human error.

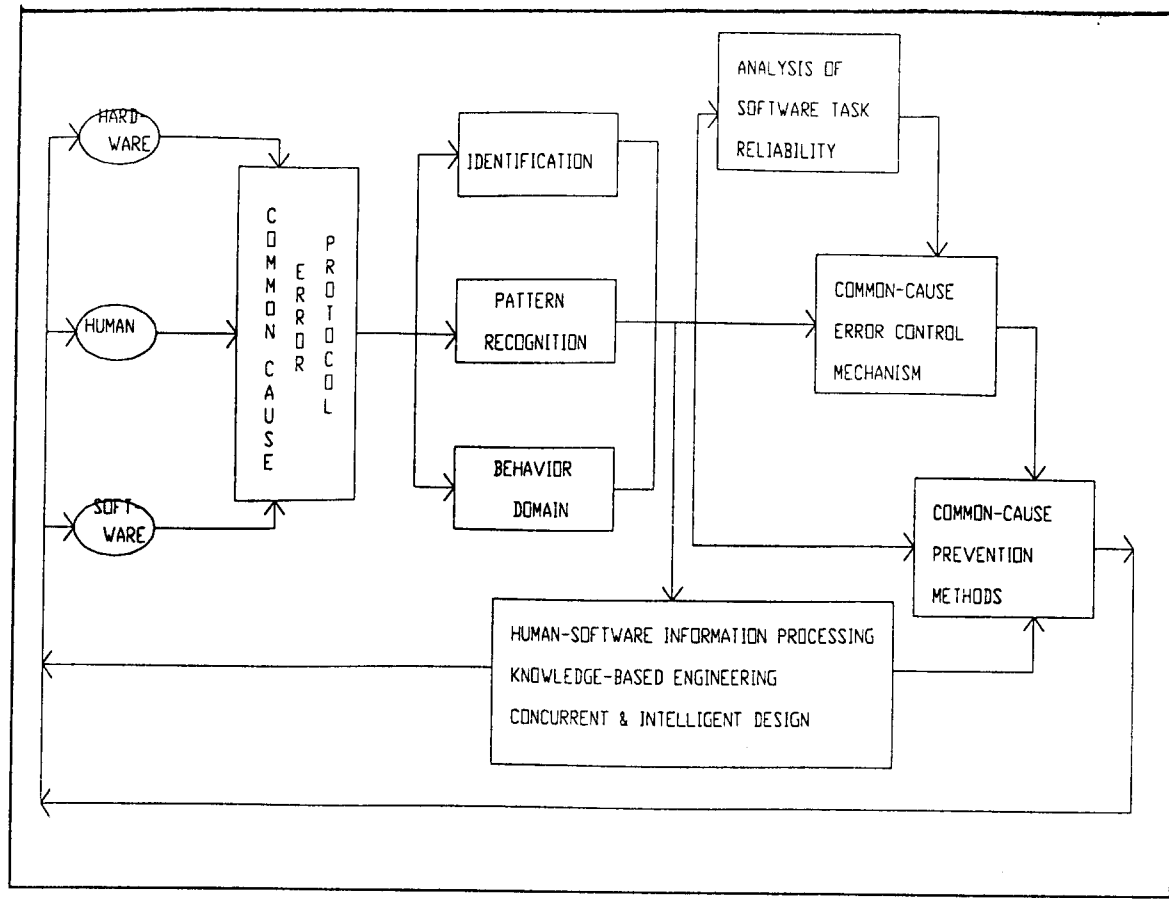

Figure 1. Schematic Design Scheme of the Common Cause Model

Three types of error were identified: skill-based, rule-based, and knowledge-based mistakes.

The mission of a specific software development project is to set up system components of human-software interaction. Each configuration is composed of a computer work station, a Central Operating Processor (COP) whose computer assigns and controls all work at the local working stations, and a Multi-Version Software (MVS) development load. One approach to software design research using such a system that tends to be expensive, is to install two independent versions of MVS developed by two completely separate software development teams/engineers. The common-cause effect affected by internal common-cause human domain errors is determined using redundant components in this case.

This study deals with the problem of common-cause human

behavior domain error in human-software interaction, that is, the major causal factors in common-cause failure effects on the multi version software development.

II. A COMMON CAUSE MODEL AND HUMAN BEHAVIOR ERROR DOMAIN

The common-cause model can be used to define internal common cause human-based error and to develop a common-cause error control mechanism for human-software interaction. It can be explained in terms of four schematic and systematic design stages, as illustrated in Figure 1. The stages are as follows:

(1) Human-software interaction components: These system components are the basic elements and factors in human-software interaction. They are: the human working as a software engineer, software as a operating system, and hardware as a system work station. The common-cause error occurs in system interactions involving failures among these system components.

(2) Common-cause error protocol: Common-cause error protocol is the actual location and identification of common-cause error attributed to a common-cause effect in a redundant system of multi-version software development. It is distinguished within a given common-cause error mode by its individual identification, by a pattern recognition, and by a behavior domain.

(3) Common-cause error function: This is the function of common cause error revealed in the existence and the performance allocation of each common-cause error mode using an evaluation typically by three variables such as frequency or error occurrence, error correction time, and point of error occurrence in time.

(4) Common-cause analysis, representation, and system redesign: This stage consists of the analysis and representation of common-

cause error in human-software interaction. Several analytical methods have been provided to define common-cause human domain error, and to redesign the system interaction with representational results and prevention schemes involved with system development productivity, and common-cause error control mechanisms.

The common-cause function is shown in the existence and in performance allocation of common-cause failure with its identification(:I_i), pattern recognition(:P_j), and behavior domain(:B_k) of common-cause error mode. Each allocated common cause error mode is evaluated by performance variables using common-cause error frequency(:F_i,j,k), error correction time(:C_i,j,k), point of error occurrence in time(:O_i,j,k) during the software development period. The common-cause function, C_r is:

C_r = C(I_i(F_i,C_i,O_i), P_j(F_j,C_j,O_j), B_k(F_k,C_k,O_k))

The common-cause function consists of these three reasoning factors of common-cause error mode, identification, pattern recognition, and behavior domain of common-cause error mode. Certain common-cause errors have these three different axes of reasoning modes, with which can be evaluated by the three subjects' performance variables, frequency, correction time, and point of occurrence in time, using the appropriate portion of the total amount of collected data relating to all errors.

Ⅲ. CONCLUSION

The new paradigm and experimental procedures showed during the study were to analyze common-causes of software development related to human behavior error domain and to identify software design factors contributing to common types of error

occurring in human-software interaction.

Therefore, characteristics and properties of new design paradigm can be applied to improving reliability of software development and to providing guidelines for design of software development.

REFERENCES

Boehm, B. W., TRW. "Improving Software Productivity." Computer, Sep. 1987, p. 43-57.

Curtis, B. "Human Factors in Software Development." IEEE, Cat. No. EHO 185-9, 1981.

Endres, Albert. "An Analysis of Errors and Their Causes in System Programs." IEEE Transactions on Software Engineering, June 1975, p. 140-149.

Mitta, Deborah. "A Methodology for Quantifying Expert System Usability." Human Factors, 1991, 33(2), p. 233-245.

Musa, J.D., A.Iannino, K. Okumoto. Software Reliability Measurement, Prediction, Application. McGraw-Hill Com., New York, 1987, p.77-101.

Park, Peom, S. K. Adams, Way Kuo. " Human Reliability and Common Cause Analysis in Software Engineering Quality Control." Proceeding of The 6th Asia Quality Control Symposium, Seoul, July, 1992, p.72-87.

Park, Peom. "Common-Cause Analysis in Human-Software Interaction: System Design, Error Control Mechanism, and Prevention." Unpublished doctoral dissertation, Iowa State U, Ames, Iowa, U.S.A., Jan. 1992.

Peters, G. "Human Error: Analysis and Control." Journal of the ASSE, Jan. 1966.

Rasmussen, J., K. Duncan, J. Leplat. "Cognitive Control and Human Error" New Technology and Human Error. John Wiley & Sons, 1987, p. 53-61.

Reason, James. "Generic Error-Modelling System(GEMS): A Cognitive Framework for Locating Common Human Error Forms." New Technology and Human Error, Ed. by J. Rasmussen, K. Duncan and J. Leplat, John Wiley & Sons. Ltd, 1987, p. 63-83.

Woods, D.D. "Modeling and Predicting Human Error." Human Performance Models for Computer-Aided Engineering, Ed. by J. I. Elkind, Academic Press, Inc., 1990.

Youngs, Edward A. "Human Errors in Programming." Human Factors in Software Development: COMPSAC81,