■ 研究論文

Journal of the Human Engineering Society of Korea
Vol.11, No.2, 1992

Causal Factors and Symptoms of Human Behavior Error Domain in Human-Software Interaction

Peom Park * · Kwan Seok. Lee **

ABSTRACT

This study is to define 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 reliabbility 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 reliable software packages.

1. 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 inde-

* Department of Industrial Engineering In Cheon University
** Department of Industrial Engineering HongIk University

pendent 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 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 usability 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, and the 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 were 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. 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 multiversion software development.

2. 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

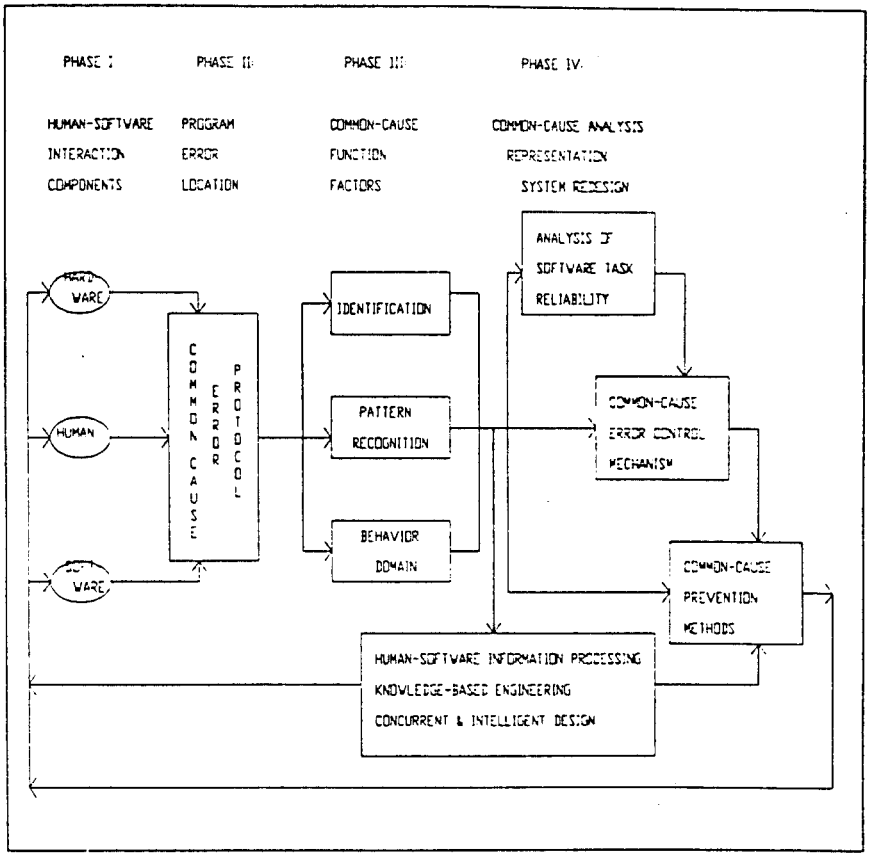

Figure 1. Schematic Design Scheme of the Common-Cause Model

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 representa-

tional 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), 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, 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.

There are three features of internal common-cause human behavior error protocol, previously introduced, that can be used in determining the identification of programming error modes, pattern recognition, and behavioral error categories of common-cause errors in human-software interaction. They are: identification of common-cause error protocol (I–i), reasoning pattern error modes (P–j), and behavior domain error modes (B–k).

There are eight identification modes (I–i) categories of typical human-based programming error from common-cause error protocols, which are used in the determination of the common-cause error that caused the failure. Each error protocol mode means the actual location of common-cause error and contributes to the common-cause effect at each stage of human-software interaction for multi-version redundant software development system(Park, 1992; Thayer, 1978).

I–1 System design and requirement errors
I–2 Variable setting and program handling errors
I–3 Program input and data base error
I–4 Computation based errors
I–5 Program logic errors
I–6 Human-software system interface errors
I–7 System operation errors
I–8 Output and output formatting errors

Common-cause reasoning patterns (P–j) can be recognized with causal characteristics which implicate the identical elements of reason, perception, control mechanism, occurrence processing, stimulus response requirement, etc. Each identical property or reason matches a pattern recognition for the common-cause human error mode(Park, 1992; Youngs, 1981; Andres, 1975).

P–1 Knowledge deficiency: There is a lack of knowledge based on the hardware system, operating system, human-software interface, field of specific requirements or problem solving methodology.

P–2 Design deficiency: During the design phase of the human software interaction system, some common-cause errors have been overlooked. These are in preliminary and detailed design work, the design reviews, definition of variables and attributes, and all work done prior to coding.

P–3 Operation and maintenance errors: Occurrence of these common cause errors may be due to improper maintenance, carelessness, or im-

proper calibration. The same program performing maintenance on all redundant units of human-software system may repeat the same mistake on all of them.

P–4 Functional deficiency: This includes misunderstanding of process variable behavior or specific requirements, inadequacy of designed protective action, inappropriate use of methods or instrumentation, or inadequacy of component processing in humansoftware interactions.

P–5 Syntax error: These result from expressions which are incorrect in the language being used regardless of the context in which they appear. Detection of these errors may be allowed through a relatively superficial analysis using grammatical rules of programming language. The programmer may detect and correct such errors as a matter of course during the programming process.

P–6 Semantic error: These occur when syntactically correct components of a program imply conditions which are untrue or impossible in stated combinations. This statement is syntactically correct, but it is impossible to allocate two different physical units to a single logical unit. These kinds of errors may require extensive analysis covering various interacting aspects and components of a program.

P–7 Logical error: These produce incorrect results but otherwise cause no obvious malfunction of the program. There is probably little which can be done in terms of redesigning compilers to aid the programmer in eliminating such errors. These errors show a lack of fit of the program to the calculation logic. Also, the program may exactly solve a different problem from the one intended.

P–8 Clerical error: These may appear to be either syntactic or semantic errors. They are only partly a function of the language used. They result from mispunched, misplaced, or miscopied cards, misread program drafts, card shuffling, or incorrect taper mounting.

P–9 System complexity: In human-software interaction systems, especially with a large-scale programming project, special difficulties arise from system components, comparing and contrasting the given requirement, the type and size of computer used, selection of proper programming language, memory size and speed required, processing time, decomposing the problem into subproblem, functions, models, and analysis. This system complexity appears to be judgmental or managerial in nature and cannot be easily defined with a lack of relation to the specific tasks of the software engineer.

There is a common-cause error category in terms of the programmer's behavioral aspects (B–k) or point of view (Figure 2.). Such common-cause error factors may be representative of errors from the human information processing, knowledge based design, error control mechanism, and human behavioral science (Park, 1992; Rasmussen, 1987; Reason, 1987). B–1 Skill-based behavior domain (B–1). A Perception and sensing; B–1. B Automated sensory-motor reaction systems):

The skill-based behavior domain is a sensory-motor pattern, controlled and automated behavior, controlled by the structure of the adaptive patterns stored in the human nervous system. It means that this human error behavior is controlled by psychological laws and physiological mechanisms governing the human software processing structure and the concept of human behavioral perception and cognition. Some characteristics of this skill-based behavior mode are as follows: (1) Sensory-motor varia-

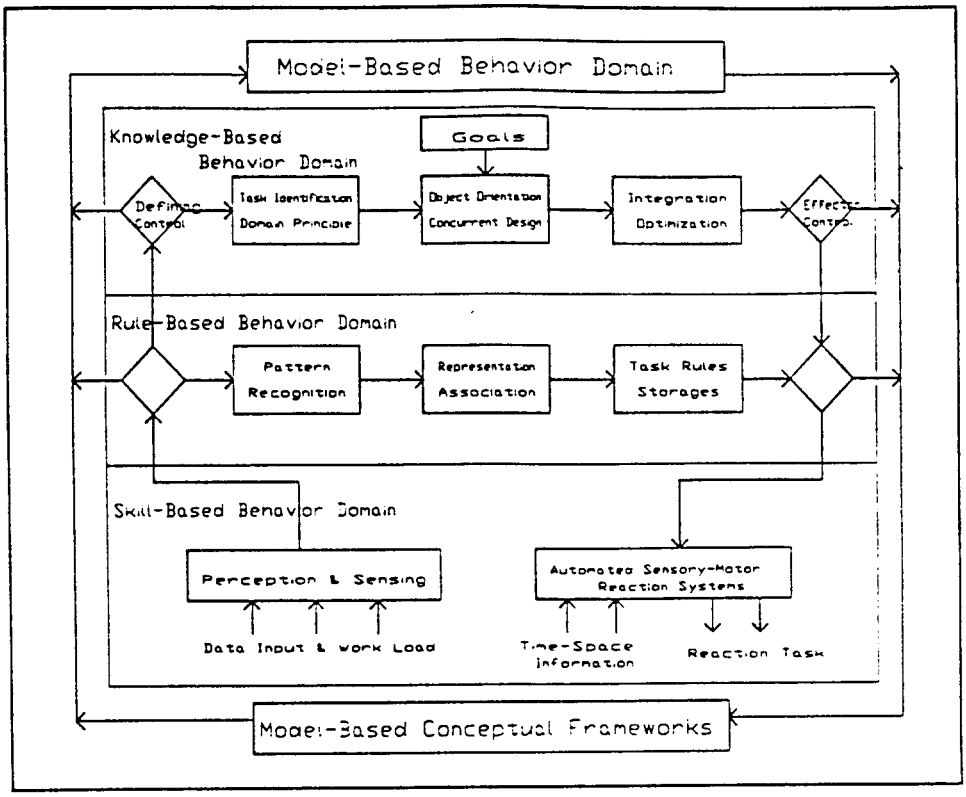

Figure 2. Human Behavior Error Domain in Human-Software Interaction

bility, (2) Recency and frequency, (3) Topographic misorientation, (4) Environmental control signal, (5) Stereotype mismatching, (6) Shared schema features, (7) Adaptation and fine tuning.

B–2 Rule-based behavior domain (B–2.A Pattern matching and recognition; B–2. B Representation and association; B–2.C Working memory and rule interpreter):

The rule-based behavior scheme is a human-software interaction that represents human reasoning with grammatical language structure and logical allocation rules. The rule-based systems represent the solution to a problem as a set of rules that specify how some string of symbols may be transformed into other strings of symbols, such as a simple form of pattern matching. The transformation of one pattern to another in a rule-based language is understood to represent an IF-THEN implication. Rules can express associations between state and task. Some characteristics of this rule-based behavior mode are as follows: (1) Habit robustness, (2) Typical fixation, (3) Availability, (4) Omission of an isolated function, (5) Over simplification, (6) Alternative mistake, (7) Over-confidence, (8) B–3 Knowledge-based behavior domain (B–3. A Task identification and domain principle; B–3. B Object orientation and concurrent design; B–3.C Integration and optimization):

The knowledge-based behavior scheme is a human behavioral phase interacting with soft-

ware development concerned with the design and implementation of programs which are capable of emulating human cognitive skills such as problem solving, task identification with domain principle, object orientation relative to the goal, concurrent design of software product and humansoftware interaction processing, and optimal system integration. The structure of the behavior is an evaluation of the situation, designing of a proper sequence of actions to achieve the goal. It depends upon fundamental knowledge of the processes, functions and anatomical structure of the system. Some characteristics of this knowledge-based behavior mode are as follows: (1) Human variability, (2) Selectivity, (3) Adaptation, (4) Working memory limitation, (5) Errors in a causal structure, (6) Availability, (7) Matching bias revisited, (8) Need for human decision making, (9) Memory cueing/reasoning by analogy, (10) Incorrect and incomplete knowledge.

B–4 Model-based behavior domain:

A highly reliable human-software interaction model yields cognitive design base strategies to define models for adaptive interface. Communication strategies for basic system design, information processing, knowledge of components, and systems configuration of interface, must be represented explicitly. The following are some adaptive concepts of model base strategies and design: symbolic and quantitative model, performance and cognitive model, static and dynamic model, syntactic and semantic model, state-transition model, singular and multiple model, etc.

3. EXPERIMENT ANALYSIS AND RESULT

After three prior pilot experiments, the main experiment was conducted with data collection. During the experiment, the contents of common-cause human error in subject programming failure were, first, recorded with an explanation of the reasons for those failures, correction time, and point of error occurrence in time. Then, at the representational interview session held every 30-45 minutes, the common-cause error protocol was allocated to each of the categorical factors: I–i, P–j, and B–k (Park, 1992). Experimental data was then validated and analyzed by statistical methods and a geometrical method using vector analysis and mapping designed for use in analyzing common cause errors in human-software reliability and interactions. Results were derived using the following analytical methods: common-cause error mode data and table, mapping and geometric vector evaluation in hexahedron contours, value of common-cause function with simulated rating, historical common-cause error recovery time zone, transition relationship diagram, grouping of major common-cause factors, and correlation and results using expert subjects was intended to identify clearly those factors related to the design of software development as distinguished from conditional factors associated with level of subject, type of language, and type of requirement. Finally, the new paradigm and the properties of common cause human behavior domain error in human-software interaction were determined by the analysis of experimental data collected on the ten expert subjects and compared with data from each of the categorical conditions.

With the error occurrence frequency factor, the major reasoning categories in each common-cause error mode are: in the identification mode, I.3 (19.4%), I.2(16.2%), and I.1 (15.9%); in the pattern recognition mode, P.2 (33.7%), P.3 (18.0%), and P.1 (15.7%); in the behavior

Table 1. Common-Cause Human Behavior Error Domain a and Experimental Data Analysis

CCM[b]	Freq	$F_{i,j,k}$[c]	CT	$C_{i,j,k}$[d]	POT	$O_{i,j,k}$[e]	CT/F[f]
I.1	54	15.9%	697.0	26.2%	369.7	70.7%	12.9
I.2	55	16.2%	317.0	11.9%	217.7	41.6%	5.8
I.3	66	19.4%	224.0	8.4%	230.7	44.1%	3.4
I.4	30	8.8%	189.0	7.1%	260.0	50.9%	6.3
I.5	38	11.2%	442.0	16.6%	320.5	61.3%	11.6
I.6	35	10.3%	354.0	13.3%	235.4	45.0%	10.1
I.7	24	7.0%	72.0	2.7%	163.3	31.2%	3.0
I.8	38	11.2%	369.0	13.8%	359.2	68.7%	9.7
Tot	340	100.0%	2664.0	100.0%	523.0[g]	51.7%[h]	7.8[i]
P.1	55	15.7%	583.5	21.2%	254.1	48.6%	10.6
P.2	118	33.7%	1234.0	44.8%	292.4	55.9%	10.5
P.3	63	18.0%	168.5	6.1%	248.8	47.6%	2.7
P.4	17	4.9%	140.5	5.1%	250.5	47.9%	8.3
P.5	24	6.8%	83.0	3.0%	223.2	42.7%	3.5
P.6	9	2.6%	80.5	2.9%	238.8	45.7%	8.9
P.7	26	7.4%	216.0	7.9%	316.9	60.6%	8.3
P.8	22	6.3%	84.5	3.1%	207.7	39.7%	3.8
P.9	16	4.6%	163.0	5.9%	198.1	37.9%	10.2
Tot	350	100.0%	2753.5	100.0%	523.8	47.4%	7.9
B.1	55	16.3%	86.5	3.2%	246.5	47.1%	1.6
B.2	123	36.5%	754.0	28.1%	257.2	49.2%	6.1
B.3	147	43.6%	1681.5	62.7%	280.0	53.5%	11.4
B.4	12	3.6%	160.5	6.0%	289.2	55.3%	13.4
Tot	337	100.0%	2682.5	100.0%	523.0	51.3%	8.0

[a] I_i : identification of common-cause error mode, P_j : pattern recognition of common-cause error mode, B_k : behavior domain of common-cause error mode.

[b] CCM : common-cause error mode.

[c] $F_{i,j,k}$: portion(%) of total frequency.

[d] $C_{i,j,k}$: portion(%) of correction time.

[e] $O_{i,j,k}$: average point of occurrence in time from total 100% completion time.

[f] CT/F : correction time per frequency (unit: time in min.).

[g] total 100% completion time.

[h] average percentage of $O_{i,j,k}$.

[i] average time of CT/F.

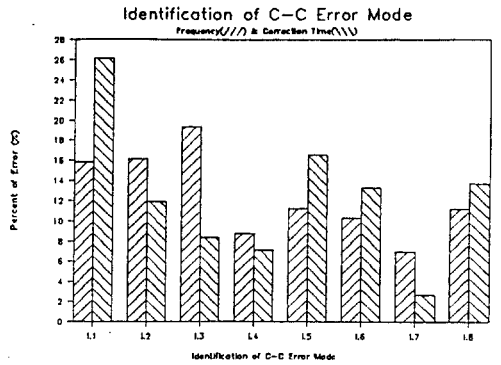

Figure 3. Portion of Identification of Common-Cause Error Mode

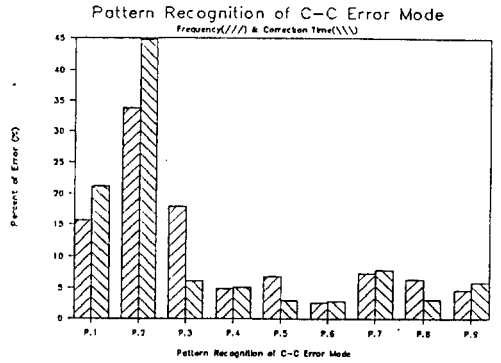

Figure 4. Portion of Pattern Recognition of Common-Cause Error Mode

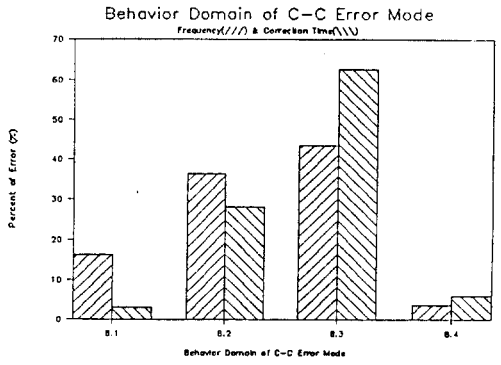

Figure 5. Portion of Behavior Domain of Common-Cause Error Mode

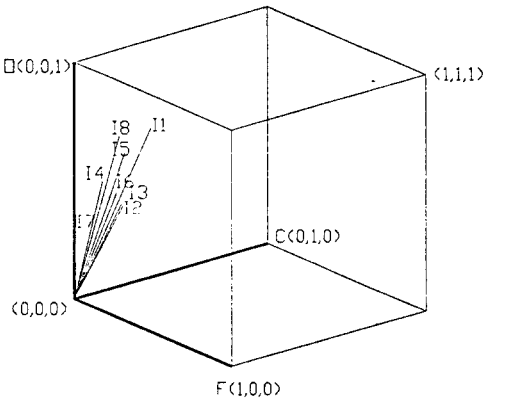

Figure 6. Identification of Common-Cause Error Mode: Geometric Configuration

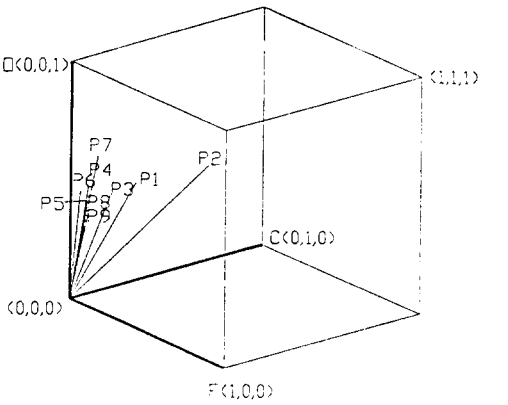

Figure 7. Pattern Recognition of Common-Cause Error Mode Geometric Configuration

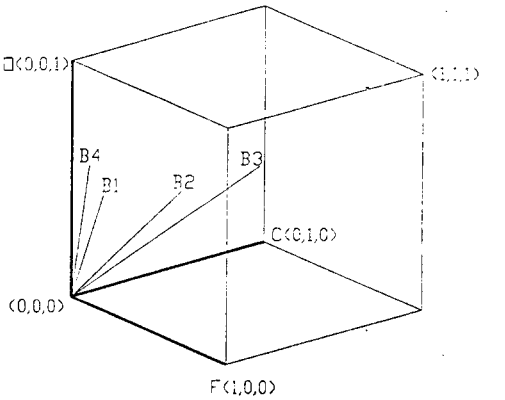

Figure 8. Behavior Domain of Common-Cause Error Mode: Geometric Configuration

domain mode, B.3 (43.6%) and B.2 (36.5%). When the error correction time factor is applied, I.1 (26.2%), I.5(16.6%), and I.8(13.9%) in the I–i mode; P.2(44.8% and P.1(21.2%) in the P–j mode; and B.3(62.7%) and B.2(28.1%) in the B–k mode (See Table 1.).

Figures 3–5 show plots of proportional mean frequency, correction time, and occurrence in time based on six criteria for characteristics in pattern recognition of common cause error mode. All trends are similar except for I.3 and I.4 in the common-cause identification mode. There are no significant differences in pattern recognition and behavior domain. However, I.4 and I.5 in the identification mode have a little difference in correction time. P.4, P.5 and P.6 in different relative proportions but the remaining common-cause error pattern recognition mode and B.4 in behavior domain mode result in different relative proportions but the remaining common-cause error modes show a strong trend for comparison among the various proportional means. There are configurations of hexahedron contour shown in Figures 6-8 which present a combined severity profile of common cause errors using each of the three factors of the common-cause function. Each common-cause mode can be evaluated by the calculation of a geometrical vector value from the geometric origin (F, C, O) = (0, 0, 0). Thus, the figure of hexahedron can be changed with different unit values on each of the three axes.

Transition related grouping of major common cause human behavior error domain factors in Figure 9 shows the transition relationships among different common-cause function modes. Common-cause properties can be grouped according to their analogical characteristics with human behavioral aspects. The heavy lines indicated more frequent transition each other, that is, more strong relationship, than the light lines. The major transit relationship group is I.3, P.3 and P.8 in B.1 Group 1; I.2, I.5, I.6, P.4, P.5, P.7 and P.9 in B.2 Group 2; and I.1, I.4, I.7, I.8, P.1 and P.2 in B.3 Group 3. The minor transit relationship group is I.7 and P.5 in B.1 Group 1; I.3, I.4, I.8, P.1, P.2, P.3 and P.6 in B.2 Group 2; I.2, I.5, I.6, P.7 and P.9 in B.3 Group 3; and I.1, I.8, P.2 and P.9 in B.4 Group 4.

Characteristic results and causal factors derived from this research are: (1) Two major common cause reasoning groups exist in human-software interaction: (a) a major group consisting of knowledge-based behavior related errors indicated by design and knowledge deficiencies; (b) another major group consisting of rule-based behavior related errors indicated by logical errors, functional deficiencies, and system complexity. (2) In training education sessions, consideration should be given to common-cause reasoning characteristics to eliminate the common-cause human error in human-software interaction. These characteristics include: (a) human-software interaction. These characteristics

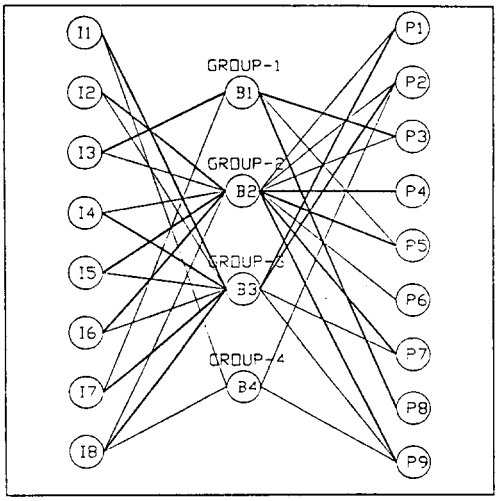

Figure 9. Transition Relationship Diagram and Common-Cause Human Domained Error Modes

include: (a) human mind robustness (pre-existing incorrect knowledge and information); (b) pattern recognition in human memory; (c) human attention and perceptual ability; (d) incompleteness of knowledge and information uncertainty. (3) Design with intelligence and concurrence by the knowledge-based processing: (a) knowledge acquisitions; (b) knowledge representation; (c) knowledge utilization.

4. CONCLUSIONS

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.

5. 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, Ed. by Bill Curtis, L.A., C.A. 1981, p. 383-392.