A Cognitive Framework for Modeling Coincident Software Faults: An Experimental Study

Authors: Bo Zhao[2], You Song[2], Wenhao Xu[2], and Fuqun Huang[1,*]

[1] Western Washington University, Department of Computer Science, Bellingham, USA

[2] Beihang University, School of Software, Beijing, China

Presenter: Fuqun Huang, PhD

Assistant Professor, huangf2@wwu.edu

SafeComp 2023

Sep. 19-22, Toulouse, France

Overview

Introduction

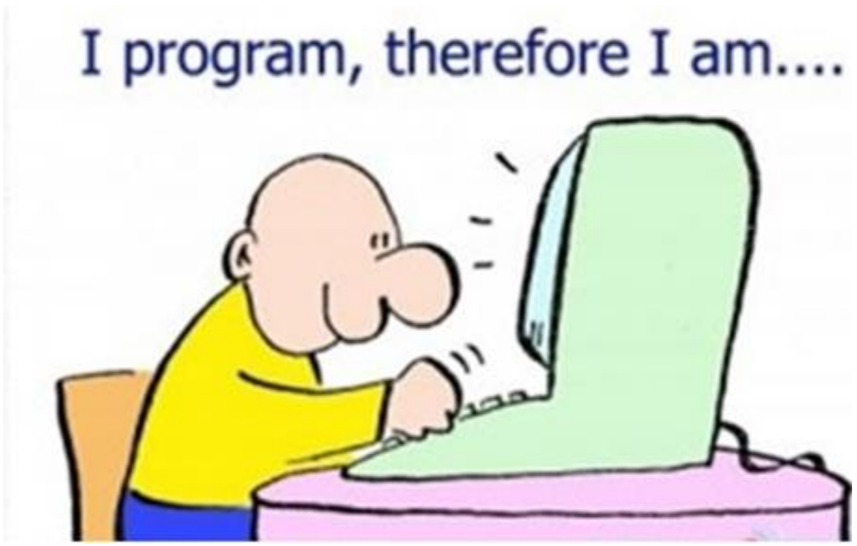
- N-version programming as a strategy for fault tolerance

  Different from Hardware?

- The degree to which a software system tolerate faults largely depends on the fault diversity between multiple versions.

Introduction (con't)

- **How to achieve software fault diversity?**
  - What's software in nature？
  - How are software faults caused?



To Err Is Human!

Introduction (con't)

- What are the human factors that contribute to software fault diversity?

Addressed today!

Concepts

- **Fault**

An incorrect or missing step, process, or data definition in a computer program.

- **Error**

An erroneous human behavior that leads to a software fault.

- **Occurrences (OC)** of a fault

The number of programmers in an N-version programming study who introduced that fault.

- **Coincident Fault**

A fault whose Occurrences is two or more in N-version programming, i.e., that was introduced by at least two programmers.

Concepts (con’t)

- **Prevalence of Occurrence (POC)**

The percentage of programmers who introduced the fault *i*, defined as:

$$POC_i = OC_i/P \qquad (1)$$

where *P* is the total number of programmers who submitted code for the task.

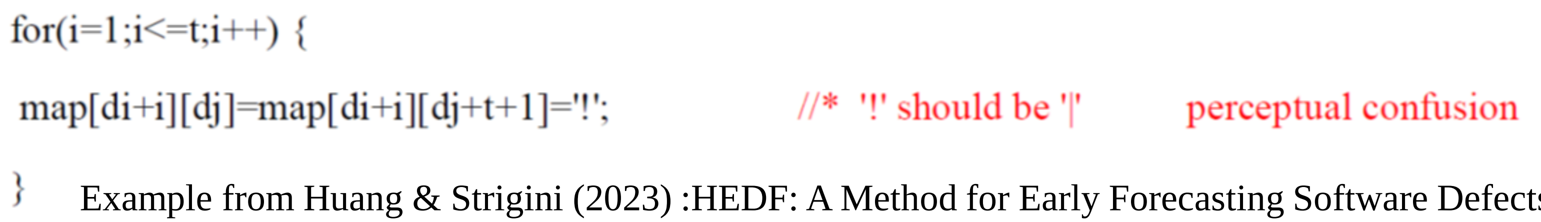
- **Larger POC$\rightarrow$ more common the fault is.**

Overview

Cognitive Framework for Coincident Faults (CognFCF)

❑ Rasmussen’s Performance Framework

- **Skill-based (SB) performance** follows from the statement of an intention, “rolls along” automatically without conscious control. Skill-based activities in programming include typing a text string, compiling a program by pressing a button in the programming environment.
- **Skill-based errors** are the human errors occurring in skill-based performances. In software development, typos and entering a wrong letter which looks similar to the correct one (e.g. taking 0 for o) are typical examples of skill-based errors In software development, typos and entering a wrong letter which looks similar to the correct one (e.g. taking 0 for o) are typical examples of skill-based errors



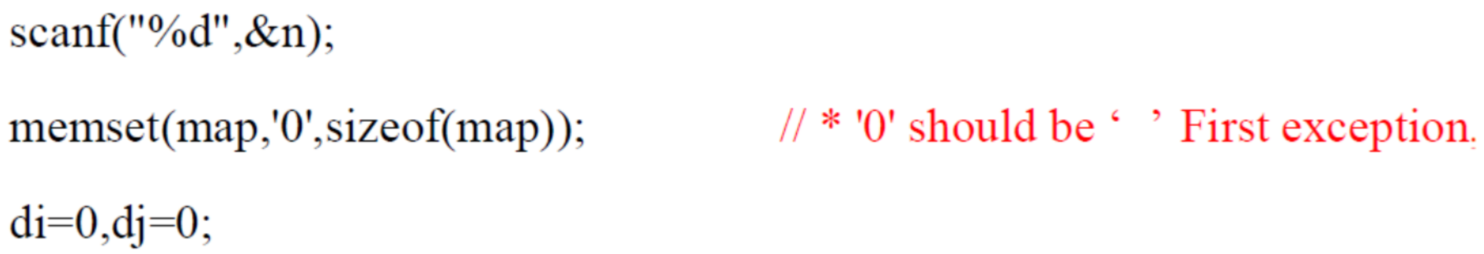
```
for(i=1;i<=t;i++) {
 map[di+i][dj]=map[di+i][dj+t+1]='!';          //* '!' should be '|'      perceptual confusion
}
```

Example from Huang & Strigini (2023) :HEDF: A Method for Early Forecasting Software Defects Based on Human Error Mechanisms, IEEE Access 11, 3626-3652

CognFCF (con’t)

❑ **Rasmussen’s Performance Framework**

- **Rule-based (RB) performance** is applicable for tackling familiar problems. It is typically controlled by stored rules that have been derived from a person’s experiences. In programming, there are many rule-based performances, such as printing of a string line, and defining a variable in one’s familiar programming language.
- **Rule-based errors** are the errors occurring in rule-based performances.



```
scanf("%d",&n);

memset(map,'0',sizeof(map));              // * '0' should be ‘  ’ First exception.

di=0,dj=0;
```

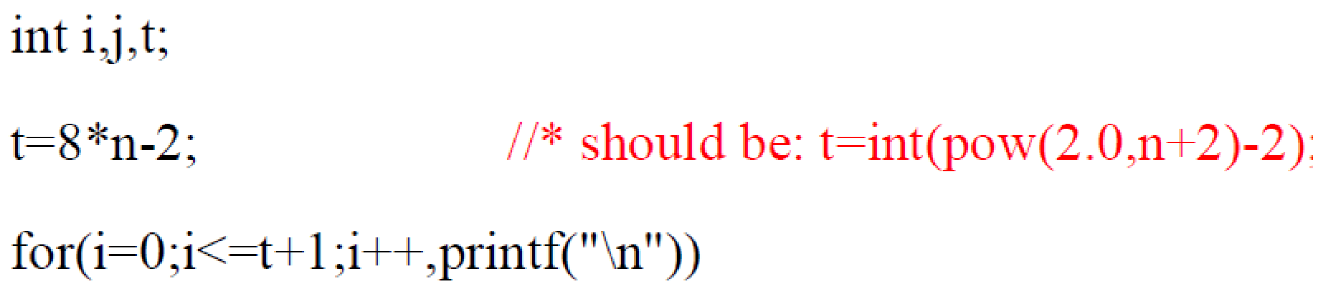
Example from Huang & Strigini (2023) :HEDF: A Method for Early Forecasting Software Defects Based on Human Error Mechanisms, IEEE Access 11, 3626-3652

CognFCF (con’t)

❑ **Rasmussen’s Performance Framework**

- **Knowledge-based (KB) performance** comes into play when individuals face novel situations, and no rules are available from previous experiences. At this level, actions must be planned using an analytical process. Errors at this level can arise from resource limitations and incomplete or incorrect knowledge.

- **Knowledge-based errors** are human errors that occur in knowledge-based performances.



```
int i,j,t;

t=8*n-2;                    //* should be: t=int(pow(2.0,n+2)-2);

for(i=0;i<=t+1;i++,printf("\n"))
```

Example from Huang & Strigini (2023) :HEDF: A Method for Early Forecasting Software Defects Based on Human Error Mechanisms, IEEE Access 11, 3626-3652

CognFCF (con't)

❑ Cross-level Errors: Post-Completion Error

Post-Completion Error

SUPPOSE Task A ={Task A.1, Task A.2};

IF <Task A.1 is the main subtask> √

AND

<Task A.2 is not a necessary condition to Task A.1> √

AND

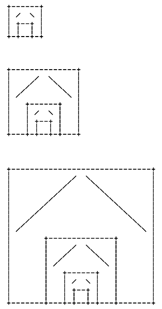
<Task A.2 is the last step of Task A >; √

THEN Humans tend to omit Task A.2.

The "Jiong" programming task

Print a series of the Chinese word "jiong" at any *n*th nested structure (1≤n≤7), and print a blank line after each word.

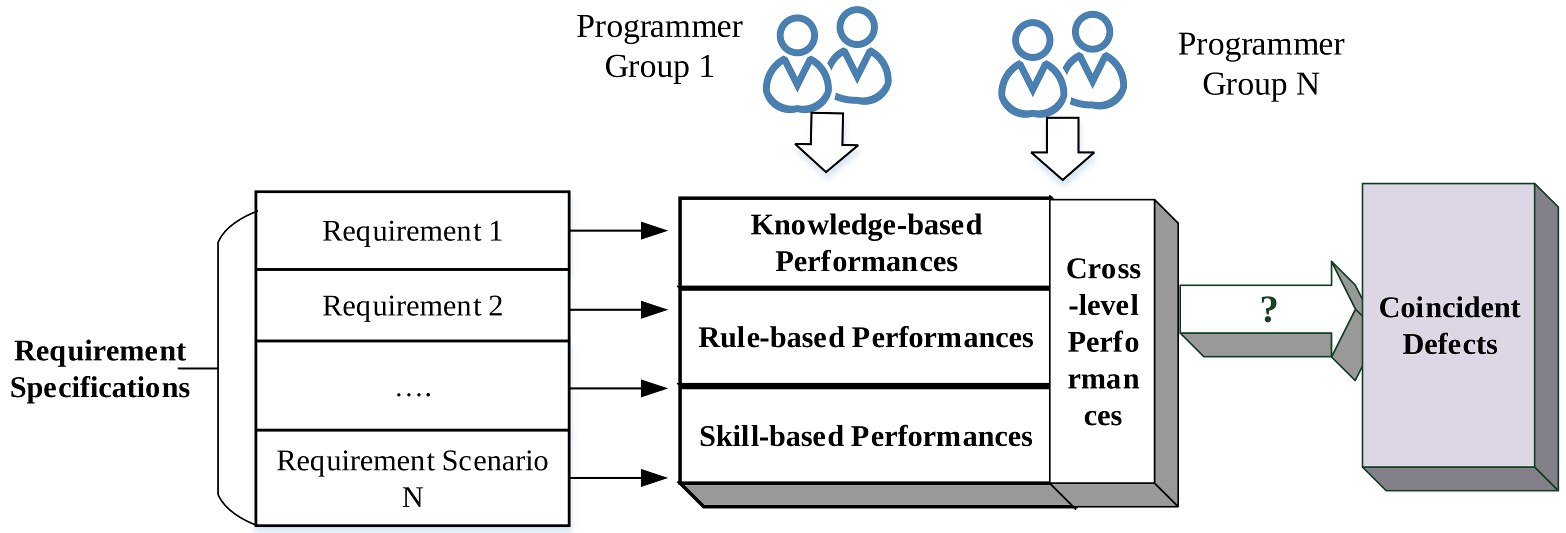
Example output for a group of three words (n=1, 2, 3, respectively):



Fault: The blank line after the "jiong" is missing

CognFCF (con't)

❑ The Cognitive Framework for Coincident Faults



Overview

Experimental Study

❑ **Research Questions (RQ) and Hypotheses (H)**

RQ 1: Do the likelihoods of a fault being a coincident fault differ across various cognitive levels?

- H1: The likelihoods of a fault being coincident are equal across the Skill-based, Rule-based, Knowledge-based, and Post-completion levels.

RQ 2: Do the Occurrences of coincident faults at skill-based, rule-based, knowledge-based, and post-completion errors differ?

- H2.1: The OC of skill-based faults is equivalent to that of rule-based faults.
- H2.2: The OC of skill-based faults is equivalent to that of knowledge-based faults.
- H2.3: The OC of skill-based faults is equivalent to that of post-completion faults.
- H2.4: The OC of rule-based faults is equivalent to that of knowledge-based faults.
- H2.5: The OC of rule-based faults is equivalent to that of post-completion faults.
- H2.6: The OC of knowledge-based faults is equivalent to that of post-completion faults.

Experimental Study (con't)

❑ **Experiment Setting**

- **Programming Task**: Bubble Sort Problem
- **The Participants:** 200 undergraduates who have completed C programming course
- **Data Collection**: Online Judge (OJ) system
- **Data Analysis**: Code inspection + Cognitive-Level Classification by two professors independent of this study

Table 1. Sample of Data

Fault ID	Fault Description	OC	CL	OJ
2	Data overflow	122	RB	WA
4	The variable is not defined before it is used.	3	RB	WA
7	Mistook the "%d\n" for "\n%d".	1	SB	PP
19	Ignored the rule that the smaller number comes first when the distances are equal.	66	PCE	WA
34	In the Bubble Sort algorithm, only the adjacent elements of the array holding the value (e.g. value[i] and value [i-1]) are exchanged, whereas the adjacent elements of the distance array are not exchanged (should be exchanged).	15	KB	WA
61	"==" is written as "=" by mistake.	11	RB	CP
62	A statement is followed by an extra '2'.	1	SB	CP
64	A "}" is missing.	5	PCE	CP
OC: Occurrences; CL: Cognitive Level; OJ: Online Judging feedback; SB: Skill-based error; RB: Rule-based error; KB: Knowledge-based error; PCE: Post-Completion Error				

WESTERN WASHINGTON UNIVERSITY

Results

- **H1: The likelihoods of a fault being coincident are equal across the Skill-based, Rule-based, Knowledge-based, and Post-completion levels.**
  - Chi-square test
  - H1 is rejected based on the results of the chi-square test on the data of the Bubble Sort Problem, χ^2 (df=3, N = 70) =22.39, p = 0.000.

Table 2. The contingency table for Chi-square test

Cognitive levels	Whether coincident		Total
	Non-coincident	Coincident	
Skill-based errors	16	0	16
Rule-based errors	10	18	28
Knowledge-based errors	13	9	22
Post-completion Errors	0	4	4
Total	39	31	70

Finding A: The proposed CognFCF has overall captured the cognitive factors underlying fault diversity, as the likelihoods of a fault being coincident at various cognitive levels in CognFCF are statistically significant different.

Results (con't)

❑ **H2.1-2.6**

Mann-Whitney test, a nonparametric test used to examine whether the means of two populations differ significantly.

OCs	Rule-based	Knowledge-based	PCE	
Skill-based	H2.1 is rejected U=368.00, N=44, p=0.000	H2.2 is rejected U=248.00, N=38, p=0.004	H2.3 is rejected U=64.00, N=20, p=0.000	**Finding B**
Rule-based		H2.4 is retained U=237.50, N=50, p=0.145	H2.5 is retained U=81.50, N=32, p=0.137	**Finding C**
Knowledge-based			H2.6 is rejected U=73.00, N=22, p=0.027	**Finding D**

Results (con't)

Finding B: The occurrences of skill-based faults are significantly lower than that of faults at any other cognitive levels. We identified a total of 16 faults at skill-based performances in our experiment; all of them were unique faults (Occurrence=1).

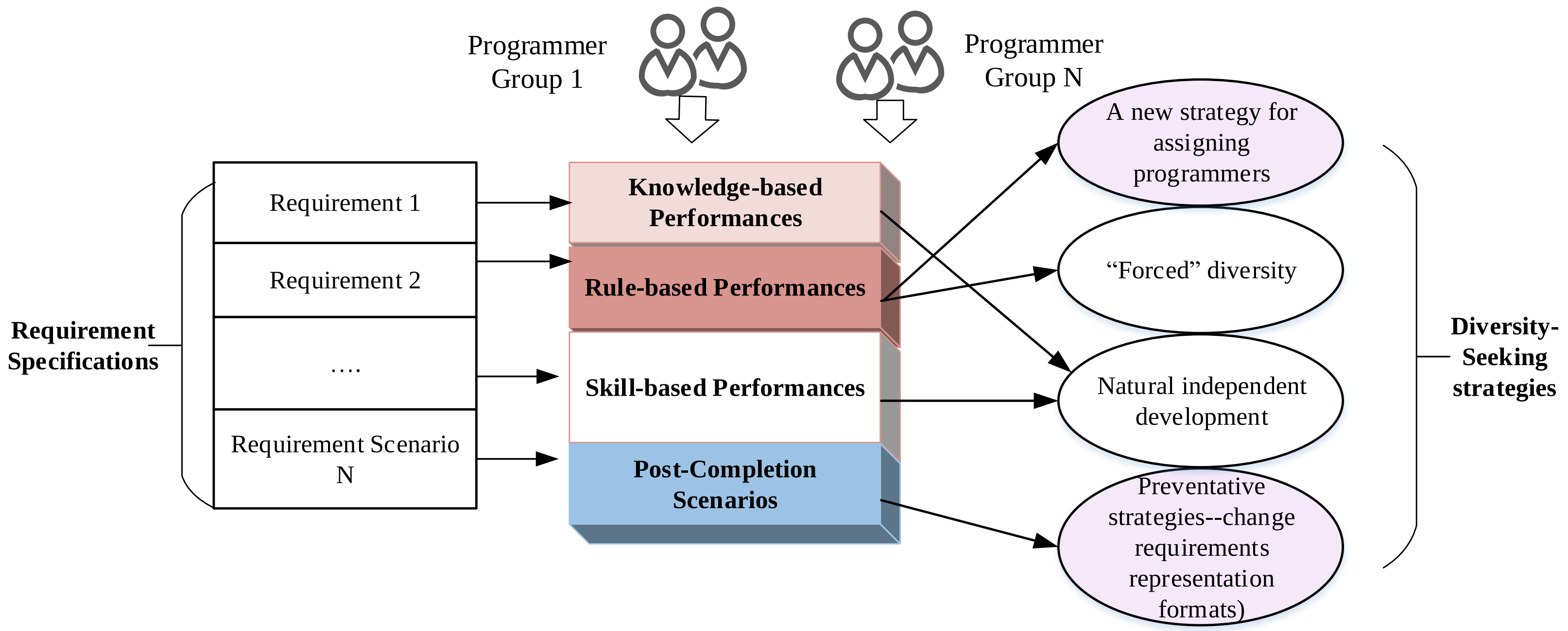
Finding C: The occurrences of rule-based faults are not significant different from that of knowledge-based and post-completion faults, however, programmers are most likely to introduce coincident faults in rule-based performances.

Finding D: Once a post-completion fault is introduced by a programmer, it is most likely to be repeated by another programmer (the conditional probability is the highest).

Discussions

❑ Implications

- Strategies for Avoiding Coincident Faults


Programmer Group 1
Programmer Group N
Requirement Specifications
Requirement 1
Requirement 2
….
Requirement Scenario N
Knowledge-based Performances
Rule-based Performances
Skill-based Performances
Post-Completion Scenarios
A new strategy for assigning programmers
"Forced" diversity
Natural independent development
Preventative strategies--change requirements representation formats)
Diversity-Seeking strategies

Conclusion

- **Contributions**
  - Proposed the first cognitive framework for modeling coincident faults
  - Designed and conducted an experimental study to validate the framework
  - Results show that the framework has overall captured the cognitive factors underlying fault diversity.
- **Future Studies**
  - Designing and evaluating the strategies for avoiding coincident faults.
  - Formally model the prevalence of errors, the relationships between different types of errors and faults.

WESTERN WASHINGTON UNIVERSITY

Thanks!

Questions?

Fuqun Huang

https://cs.wwu.edu/huangf2

huangf2@wwu.edu