A Taxonomy of Software Defect Forms for Certification Tests in Aviation Industry

Authors: Fuqun Huang[1], Bing Huang[2], Yiqun Wang[3], and Yichen Wang[4]

[1] Western Washington University, Department of Computer Science, Bellingham, USA
[2] Nanyang Technological University, School of Computer Science and Engineering, Singapore
[3] Antares Testing LLC, Beijing, China
[4] Beihang University, School of Software, Beijing, China

Presenter: Fuqun Huang, PhD

Assistant Professor, huangf2@wwu.edu

SafeComp 2023

Sep. 19-22, Toulouse, France

WESTERN WASHINGTON UNIVERSITY

Outline

- Motivation
  - Definitions
    - Methodologies
    - The Taxonomy
  - Validation and Assessment
- Discussions

WESTERN WASHINGTON UNIVERSITY

Motivation (con't)

- A good defect classification is necessary for every organization
  - Necessary for communication between developers, code reviewers, quality assurance engineers
  - Process improvement, e.g., defect prevention

Motivation (con't)

- **What is a "good" defect taxonomy**

  - Functionality: capture "what's wrong" with a piece of code or document
  - Inter-coder reliability: the degree to which different people classify a defect into the same category

- **What features determine inter-coder reliability**

  - Orthogonality ↑
  - Complexity ↓

Motivation (con't)

- **A practical need in the Chinese Aviation industry**
  - A rigorous process for assuring software quality
  - Software certification tests are conducted by qualified and independent third-party software testing centers
  - A defect taxonomy that should be easily understood and made consensus between multiple stakeholders, e.g. user representatives, developers, project managers, domain experts, and software certification testing engineers.
- **Current classification**
  - Documentation, Program, Design, and Others.
- **Why not the widely reported classifications, e.g. ODC?**

WESTERN WASHINGTON UNIVERSITY

Definitions

- **Defect**

An incorrect or missing step, process, or data definition in software, including computer programs and documentation (adapted from the definition of “fault” in the IEEE Standard Glossary of Software Engineering Terminology [8]).

- **Defect Form (DF)**

The way how a snippet of software (including computer programs and documentations) is being a defect. Note that the Defect Form here is not the same as the “Defect Type” concept commonly used in software engineering, e.g. in “Orthogonal defect classification” [1]. Defect Form describes “what” a defect is, in more detail than “Defect Type”.

WESTERN WASHINGTON UNIVERSITY

Outline

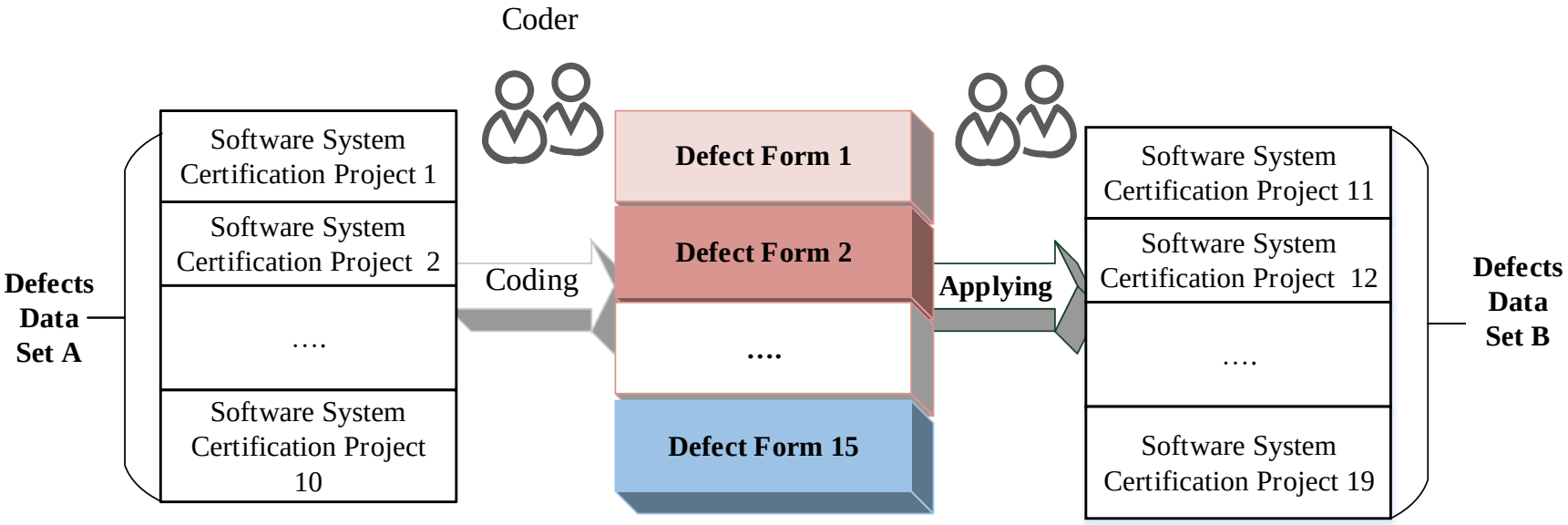
- Motivation
  - Definitions
    - Methodologies
    - The Taxonomy
  - Validation and Assessment
- Discussions

Methodologies

- **Predetermined vs. Finding patterns in real data**
- **Grounded Theory**
  - Grounded Theory, initially developed by Glaser and Strauss [9], is a methodology used to build new theories based empirical data. The methodology involves a set of strategies
  - Inductive and iterative process to generate a new theory from empirical data
  - Open Coding

Methodologies (con't)

- **The Process**



Methodologies (con't)

- **The Data**

#	Software	LOC	# Defects found in certification tests			
			Critical	Major	Regular	Minor
1	Flight management CPU	75572	0	1	5	0
2	Flight Management MIO	9692	1	0	6	0
3	Inertial Navigation Software	26406	0	1	13	0
4	Inertial Attitude Software	18361	0	3	1	2
5	Anti-jamming all-in-one	21683	0	0	6	9
6	Radio Altimeter Software	7572	1	2	0	0
7	Task management software	20353	0	2	35	3
8	Power management	21947	0	4	3	0
9	Integrated monitoring	49450	0	0	18	3
10	Portable Maintenance Aid	27314	0	3	2	0
Total		278350	2	16	89	17
			124			

WESTERN WASHINGTON UNIVERSITY

Outline

- Motivation
  - Definitions
    - Methodologies
    - The Taxonomy
  - Validation and Assessment
- Discussions

The Taxonomy

- **Thirteen Defect forms**

Defect form	Number of defects	Percentage
DF6-Inconsistency between program function and requirements specification (II) (fixed by changing the program)	35	28%
DF5-Inconsistency between requirements specification and program function (I) (fixed by changing requirements documents)	19	15%
DF1-Useless requirements specification	13	10%
DF9-Algorithm error	11	9%
DF12-Exception handling Error	10	8%
DF11-Missing function	10	8%
DF8-Calculation error	9	7%
DF2-Missing requirements specification	8	6%
DF3-Inappropriate organization of the requirement specifications	2	2%
DF10-Assignment error	2	2%
DF13-Annotation error	2	2%
DF7-Ambiguous requirements specification	2	2%
DF4-Incorrect requirements specification	1	1%
Total	124	100%

WESTERN WASHINGTON UNIVERSITY

Outline

- Motivation
  - Definitions
    - Methodologies
    - The Taxonomy
  - Validation and Assessment
- Discussions

Validation

- **Participants**

5 experienced software certification testing engineers, all of whom held team leader positions

- **Procedures**

  - The participants used the defect form list to reclassify the defects they had found in the projects they were actively involved in.
  - We also added another category, DF14-Others, to the list to identify any defects that could not be classified using the 13 forms.
  - They were encouraged to raise any issues if there were any unclear definitions or any defects that could not be classified by the taxonomy.
  - We asked them whether they had encountered any defect that could be classified by more than one defect form to evaluate the degree of mutual exclusivity between the defect forms.

Validation (con't)

- **The Validation Data**

Software systems	Critical	Major	Regular	Minor	Total
CJGZKZ	4	0	6	0	10
JZCL	5	0	0	0	5
ZKGL	9	0	16	6	31
TXCL	2	0	5	0	7
HJGL	1	0	13	1	15
BFCC	3	0	2	1	6
IISS	0	4	1	2	7
DXX	5	0	5	5	15
JT	7	0	63	3	73
Total	36	4	111	18	169

Validation (con't)

- **The Validation Results**

Defect Forms	Doc	Prog.	Other	Total #	#%
DF2	3	-	-	3	1.8
DF5	64	-	-	64	37.9
DF6	2[a]	38	-	40	23.7
DF8	-	2	-	2	1.2
DF9	-	4	-	4	2.4
DF10	-	30	-	30	17.8
DF11	-	8	-	8	4.7
DF12	-	9	-	9	5.3
DF13	-	-	6	6	3.6
DF14	1	2	-	3	1.8
Total	70	93	6	169	100

[a.] Comments in the program are inconsistent with requirements

Assessment

- **Completeness**

$$Completeness = \frac{\sum_1^{13} DF_i}{\sum_1^{14} DF_i} \times 100\% \qquad (1)$$

where DF_{14} represents a defect was assigned to “other”. That is, Completeness is estimated based on the number of defects assigned to the 13 defined forms out of all defects.

  - The taxonomy effectively covered 98% (166/169) of defects identified in the nine software certification projects.
  - The participants utilized only 9 out of the 13 available defect forms

- **Clarity and non-overlapping**

  - No issue was raised on these matters by the participants.

Discussions

- **Contributions**

- This paper proposed a new concept “defect form” to describe the patterns of defects found in certification tests in the Chinese aviation industry.
- We developed a taxonomy consisting of 13 defect forms derived from 10 software systems using Grounded Theory.
- The taxonomy were applied and validated by 5 independent professional certification testing engineers on another 9 software systems certification projects.
- Results show that nine defect forms were able to describe 98% defects.

WESTERN WASHINGTON UNIVERSITY

Discussions

Fuqun Huang

https://cs.wwu.edu/huangf2

huangf2@wwu.edu