Journal of Robotics, Networking and Artificial Life
Vol. **6**(**3**); December (2019), *pp.* 143–147
DOI: https://doi.org/10.2991/jrnal.k.191202.001; ISSN 2405-9021; eISSN 2352-6386
https://www.atlantis-press.com/journals/jrnal

Research Article

Application of Pairwise Testing into BWDM which is a Test Case Generation Tool for the VDM++ Specification

Tetsuro Katayama[1,*], Futa Hirakoba[1], Yoshihiro Kita[2], Hisaaki Yamaba[1], Kentaro Aburada[1], Naonobu Okazaki[1]

[1]*Department of Computer Science and Systems Engineering, Faculty of Engineering, University of Miyazaki, 1-1 Gakuen-kibanadai nishi, Miyazaki 889-2192, Japan*
[2]*School of Computer Science, Tokyo University of Technology, 1404-1 Katakuramachi, Hachioji City, Tokyo 192-0982, Japan*

ARTICLE INFO

Article History
Received 15 October 2018
Accepted 22 November 2018

Keywords
Software testing
boundary value analysis
pairwise testing
formal methods
VDM++
PICT

ABSTRACT

Verification tool for Vienna Development Method (BWDM) is a test case generation tool for the VDM++ specification. The existing BWDM could cause a combinatorial explosion of the generated test cases. To reduce the number of test cases, there is Pairwise Independent Combinatorial Testing Tool (PICT): a pairwise testing tool. We apply pairwise testing into BWDM. Here, BWDM cannot call PICT library directly. Hence, we have developed PICT-wrapper. It is an interface to connect PICT and BWDM. The extended BWDM eliminate the possibility of the combinatorial explosion.

1. INTRODUCTION

As the importance of software in society is increasing, specifications using specification languages become more important [1]. It is necessary to test the developed software, but it takes much time and effort to design test cases manually. So, we developed Verification tool for Vienna Development Method (BWDM) [2,3].

In BWDM, test cases are generated by boundary value analysis and symbolic execution from VDM++ specification. However, it is possible to cause a combinatorial explosion by the number of the test cases because the existing BWDM generate by all combinations when boundary value analysis. There is a pairwise testing [4] as an effective testing method to reduce the total number of the combinations. According to Kuhn et al. [4], there are very few defects occurring in three or more factors. Therefore, combination testing is effective for two factors. (To find the defects occurring in three or more factors, you should test with other techniques.) And, to test combinations of only two factors is called pairwise testing. In order to solve the above problem in the existing BWDM, this research extends BWDM so that the pairwise testing can be applied to boundary value analysis in BWDM. In applying the pairwise testing, we use Pairwise Independent Combinatorial Testing Tool (PICT) developed by Microsoft Corporation [5]. We have developed PICT-wrapper. It is an interface to connect PICT and BWDM. And, we have extended BWDM in that PICT-wrapper is embedded.

*Corresponding author. Email: kat@cs.miyazaki-u.ac.jp

2. THE EXISTING BWDM

BWDM has the following features:

- Test cases generation by symbolic execution [2].
- Test cases generation by boundary value analysis [3].

In symbolic execution, it is expected that the test cases generated by symbolic execution can cover the all execution flow. Test cases generated by boundary value analysis can be used for boundary testing. By using BWDM, efficiency of software development for VDM++ specification can be improved.

However, there is a problem with BWDM. In generating test cases by boundary value analysis, the number of the generated test cases is determined by multiplying values of each that the factors can take. For example, in the case of factors (6, 6, 2, 4, 5, 7), BWDM generates 10,080 test cases. That is, a combination explosion may occur. In this paper, we extend BWDM to solve this problem.

3. THE EXTENDED BWDM

We have developed PICT-wrapper. It is an interface to connect PICT and BWDM. And, we have extended BWDM in that PICT-wrapper is embedded. The details are described below.

3.1. PICT-wrapper

Although PICT is a Command Line Interface (CLI) tool, PICT library, which can be used from C++ programs, is provided as

an Application Programming Interface (API). However, since the existing BWDM is written in Java, it cannot call the PICT library. Therefore, as a preparation for extension of BWDM, we develop another PICT library that can be called from Java programs using Java Native Access (JNA) [6]. We call it PICT-wrapper.

Figure 1 shows the class diagram of the PICT-wrapper. Each class is explained below.

- Pict class - A PICT library written in C++ developed by Microsoft.
- LibPict class - PICT library interface written in Java using JNA. All method names are the same as the function names of the PICT library. The functions mainly used in the PICT-wrapper are explained below.
  - PictAddParameter - Registering factors and levels to PICT.
  - PictGenerate - Generating combination data.
  - PictGetNextResultRow - Getting generated data.
- PictWrapper class - A class for operating PICT. It is written in Kotlin [7]. The methods are explained below.
  - createTask - Creating and initializing a task. The task is a unit of PICT combination generation processing. It corresponds to PictCreateTask in the PICT library.
  - setRootModel - Registering a model in the task. The model is a set of factors. It corresponds to PictSetRootModel in the PICT library.
  - generate - Generating a combination using the pairwise testing. It corresponds to PictGenerate in the PICT library.
- Model class - A class for holding a set of factors. It is written in Kotlin. Its constructor and the method are explained below.
  - constructor - Generating a model in the PICT library. It corresponds to PictCreateModel.
  - addFactor - Registering Factor class to the model. It corresponds to PictAddParameter in the PICT library.
- Factor class - This class represents a factor. It is written in Kotlin. The member variables are explained below.
  - level - A level of an input variable.
  - named_level - A set of possible values of a factor.
  - *n* - The minimum number of pairs (two by default).
  - weights - A set of weights of possible values of a factor.
  - name - Name of the factor.

By changing the value of the member variable *n*, input data covering *n* combinations can be generated for the factor.

3.2. Application of Pairwise Testing

Figure 2 shows the flow of the extended BWDM. In the boundary value analysis unit, it extracts boundary values of minimum and maximum values of types for each argument accordance with conditional expression in the function of VDM++ specification. In the existing BWDM, all combinations of boundary values generated

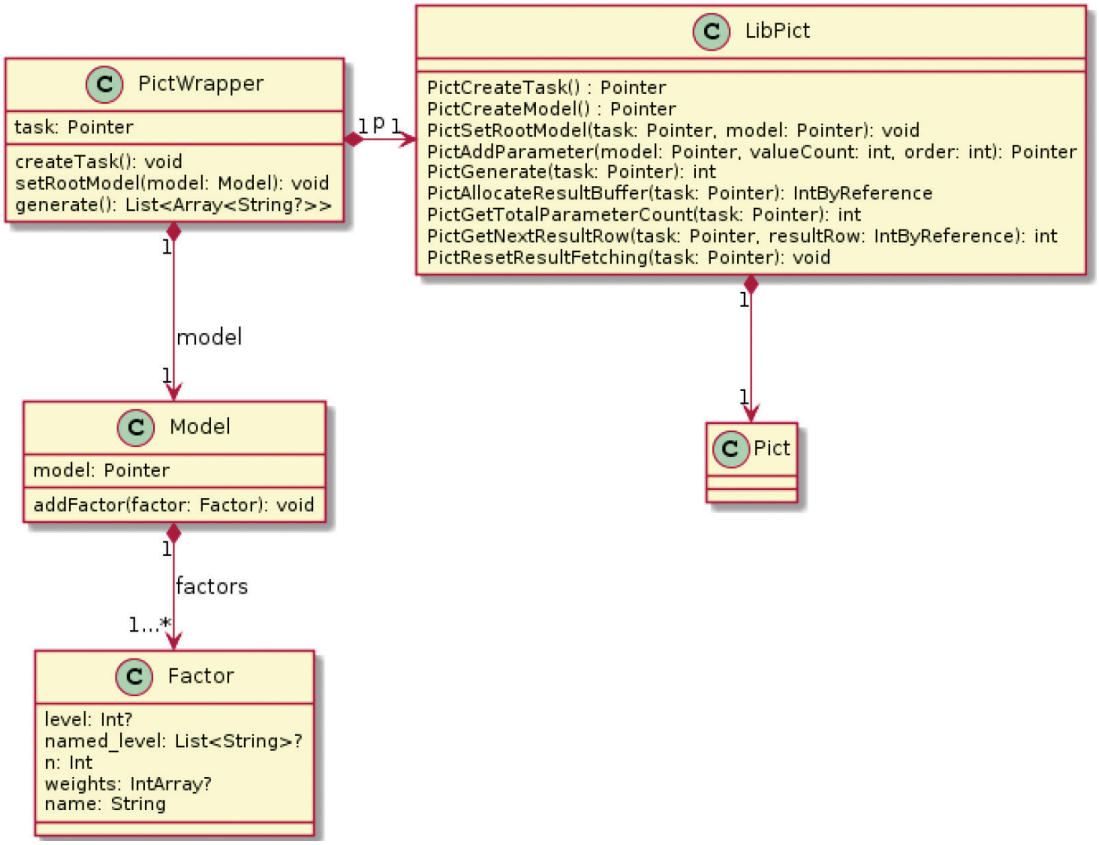

Figure 1 | Class diagram of PICT-wrapper.

for each argument are generated as input data in boundary value testing.

In the extended BWDM, input data are generated by applying the pairwise testing using the PICT-wrapper described in Section 3.1. Specifically, it inputs parameters that can be taken by factor obtained by boundary value generation into the PICT-wrapper. In the pairwise testing, by focusing on a pair of two factors, a combination testing of all pairs is executed. Here, in the PICT-wrapper by changing its setting, it is also possible to make N pairs of factors.

The input data generation algorithm of the PICT-wrapper that received boundary value data is explained below. The flow of the PICT-wrapper using this algorithm is shown in Figure 3.

PICT-wrapper generates test data, using factors and values of each factor, which are given by boundary value analysis result, as arguments.

i. By using PictAddParameter, register factors and levels in PICT.

ii. By using PictGenerate, generate combination data to which the pairwise testing was applied.

iii. By using PictGetNextResultRow, acquire one combination data generated by step ii. The acquired data is an index of parameters that can be taken by the factors.

iv. Generate output data of PICT-wrapper using parameters corresponding to the index of data acquired by step iii.

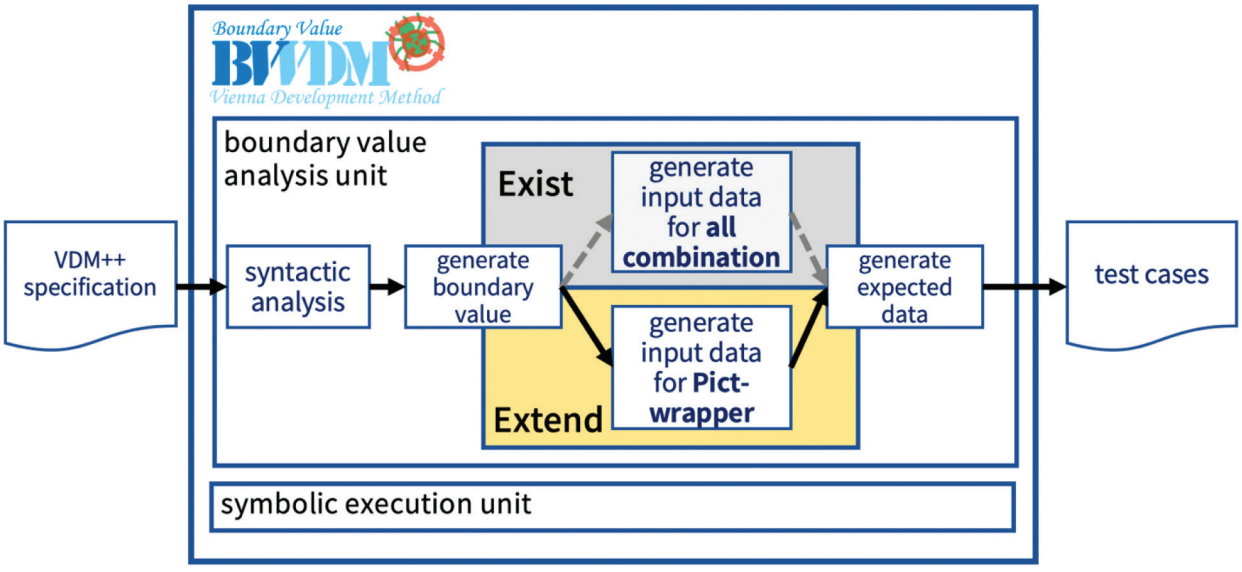

Figure 2 | The flow of the extended BWDM.

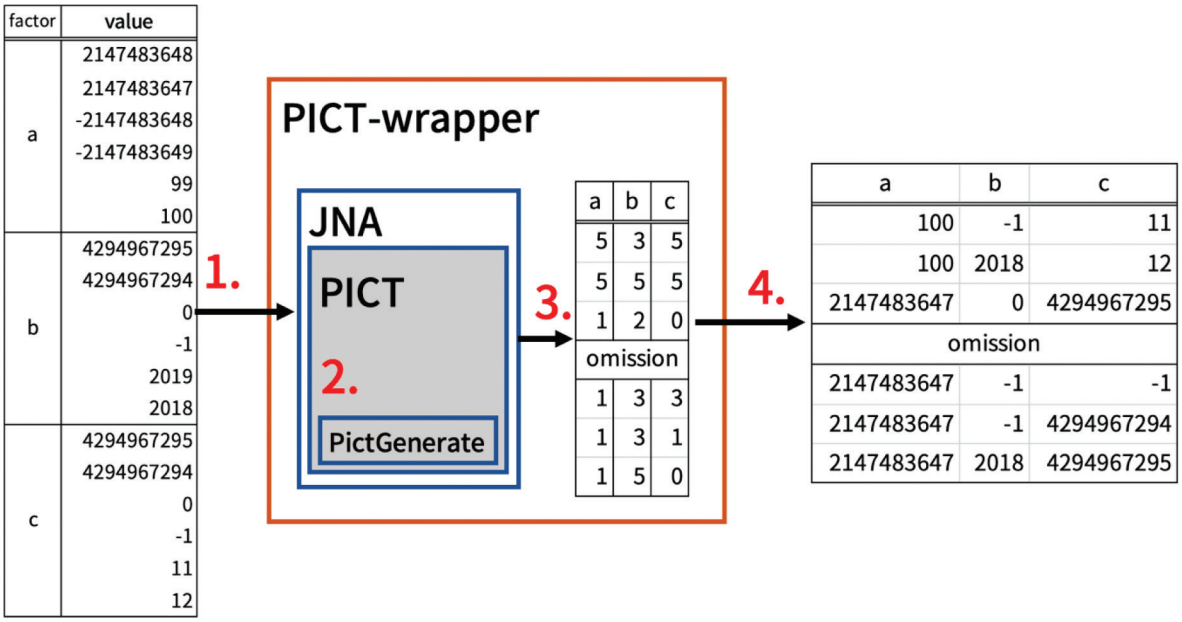

Figure 3 | The flow of PICT-wrapper.

4. APPLICATION EXAMPLE

The results of applying a VDM++ specification to the extended BWDM is shown in Table 1. In the VDM++ specification, the factor is 3, the level is a function of (6, 6, 6).

In the existing BWDM, 216 test cases were generated. In the extended BWDM, 40 test cases are generated test cases. The number of the generated test cases can be reduced. In addition, it was confirmed that 40 test cases can cover all combinations of pairs of two factors.

5. EVALUATION

We confirm that the extended BWDM can reduce the number of test cases compared with existing BWDM. In the experiment of the comparison, we input function of factor 7 and level (6, 8, 6, 8, 8, 6, 6) into BWDM. A comparison of generation results is shown in Table 2. The execution environment is macOS 10.13.6 (CPU: Intel Core i5 2.3 GHz, RAM: 16 GB). The formula used for comparison is shown below.

$$\text{Reduction rate (\%)} = \frac{A - B}{A} \times 100 \tag{1}$$

where

A is total test cases generated by the existing BWDM.

B is total test cases generated by the extended BWDM.

From Table 2 and Equation (1) the number of generated test cases could be reduced by (663,552 − 78)/663,552 × 100 = 99.98(%). The existing BWDM generated a huge number of test cases. But, the extended BWDM generated a practical number of test cases. Therefore, the extended BWDM can eliminate possible to cause a combination explosion of test cases. Furthermore, test case generation time could be shortened (see Table 2). As a result, the extended BWDM is more highly practical.

However, the extended BWDM cannot use a part of features of PICT library. For example, the extended BWDM cannot set a constraint of a specific combination. The restriction of the specific combination means a combination which must be tested or a combination which does not need to be tested. In order to conduct flexible testing, it is necessary to deal with it in future. Also, since BWDM still has few VDM++ syntax that can be applicable to it, it is necessary to expend its applicable range. In the case of A boundary value analysis for the extended BWDM, it cannot analyze conditional expression that contains two or more variables. By solving these problems, BWDM will become more practical.

Table 1 | Application result

Number	Input			Expected output
1	100	−1	11	Undefined action
2	100	2018	12	"a is 100 or more and c is 12 or more"
3	2, 147, 483, 647	0	4, 294, 967, 295	Undefined action
~	~	~	~	~
38	2, 147, 483, 647	−1	−1	Undefined action
39	2, 147, 483, 647	−1	4, 294, 967, 294	Undefined action
40	2, 147, 483, 647	2018	4, 294, 967, 295	Undefined action

Table 2 | Comparison of the generation result

	Number of generated test cases	Execution time (s)
The existing BWDM	663,552	6.767146152
The extended BWDM	78	0.88973152

6. CONCLUSION

In this paper, we have extended BWDM so that the pairwise testing can be applied to boundary value analysis in BWDM. The purpose is to eliminate the possibility of the combinatorial explosion occurring in the number of test cases generated from boundary value analysis results. The extended BWDM performs boundary value analysis on the VDM++ specification, applies a pairwise testing, and automatically generates test cases. As an evaluation result, we confirmed the extended BWDM can eliminate possibility to cause the combination explosion of the test cases generated from the boundary value analysis result. Therefore, the extended BWDM is more highly practical. Also, the extended BWDM is expected to improve the efficiency of the test process because the test cases for a function including many factors and many levels have become a practical number.

The future issues are as follows.

- Expanding an applicable range of BWDM.
- Corresponding to a constraint of the specific combination.
- Corresponding to boundary value analysis of conditional expression including two or more variables.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

REFERENCES

[1] A.E. Haxthausen, An introduction to formal methods for the development of safety-critical applications, DTU Informatics, Technical University of Denmark, 2010, pp. 1–32.

[2] H. Tachiyama, T. Katayama, Structure recognition method of if-then-else expression in BWDM using VDM++ specification, IPSJ/SIGSE Software Engineering Symposium (SES2017), 2017, pp. 130–137 (in Japanese).

[3] H. Tachiyama, T. Katayama, Y. Kita, H. Yamaba, K. Aburada, N. Okazaki, Prototype of test cases automatic generation tool BWDM based on boundary value analysis with VDM++, Proceedings of the 2017 International Conference on Artificial Life and Robotics, Seagaia Convention Center, Miyazaki, Japan, 2017, pp. 275–278.

[4] D.R. Kuhn, D.R. Wallace, A.M. Gallo, Software fault interactions and implications for software testing, IEEE Trans. Softw. Eng. 30 (2004), 418–421.

[5] Microsoft Corp. Microsoft/pict Pairwise Independent Combinatorial Testing. Available from: https://github.com/Microsoft/pict (last accessed on August 17, 2019).

[6] java-native-access/jna, Java Native Access. Available from: https://github.com/java-native-access/jna (last accessed on August 17, 2019).

[7] JetBrains: Kotlin Programming Language. Available from: https://kotlinlang.org/ (last accessed on August 17, 2019).

Authors Introduction

Tetsuro Katayama

Tetsuro Katayama received the PhD degree in engineering from Kyushu University, Fukuoka, Japan in 1996. From 1996 to 2000 he has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2000 he has been an Associate Professor at Faculty of Engineering, Miyazaki University, Japan. He is currently a Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include software testing and quality. He is a member of the IPSJ, IEICE, and JSSST.

Hisaaki Yamaba

Hisaaki Yamaba received the B.S. and M.S. degrees in chemical engineering from Tokyo Institute of Technology, Japan, in 1988 and 1990, respectively, and the PhD degree in systems engineering from University of Miyazaki, Japan, in 2011. He is currently an Assistant Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include network security and user authentication. He is a member of SICE and SCEJ.

Futa Hirakoba

Futa Hirakoba received the Bachelor's degree in engineering (computer science and systems engineering) from University of Miyazaki, Japan in 2018. He is currently a Master's student in Graduate School of Engineering at the University of Miyazaki, Japan. His research interests include test case generation and formal specifications.

Kentaro Aburada

Kentaro Aburada received the B.S., M.S., and PhD degrees in computer science and system engineering from the University of Miyazaki, Japan, in 2003, 2005 and 2009, respectively. He is currently an Associate Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include computer network and security. He is a member of IPSJ and IEICE.

Yoshihiro Kita

Yoshihiro Kita received a PhD degree in systems engineering from University of Miyazaki, Japan, in 2011. He is currently an Assistant Professor with the Computer Science, Tokyo University of Technology, Japan. His research interests include software testing and biometric authentication.

Naonobu Okazaki

Naonobu Okazaki received his B.S., M.S., and PhD degrees in electrical and communication engineering from Tohoku University, Japan, in 1986, 1988 and 1992, respectively. He joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation in 1991. He is currently a Professor with the Faculty of Engineering, University of Miyazaki since 2002. His research interests include mobile network and network security. He is a member of IPSJ, IEICE and IEEE.