SUGISAKA MASANORI

Journal of Advances in Artificial Life Robotics
Vol. 4(3); June (2025), pp. 167–171
ON LINE ISSN 2435-8061; ISSN-L 2435-8061
https://alife-robotics.org/jallr.html

Research Article

Extending BWDM to Support Various Types and Recursive Definitions in VDM++ Test Case Generation

Shota Takakura[1], Tetsuro Katayama[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]Department of Information Security, Faculty of Information Systems, Siebold Campus, University of Nagasaki, 1-1-1 Manabino, Nagayo-cho, Nishi-Sonogi-gun, Nagasaki, 851-2195 Japan

ARTICLE INFO

Article History

Received 24 November 2023

Accepted 28 June 2024

Keywords

Software testing
Formal methods
VDM++
Automatic generation
Test cases

ABSTRACT

Generating test cases from VDM++ formal specifications, which help to eliminate ambiguities, is both time-consuming and labor-intensive. To solve this problem, our laboratory has developed BWDM, a tool for automatic test case generation from VDM++ specifications. However, the original BWDM only supports integer types and cannot handle test cases for operations and functions with recursive structures. To enhance BWDM's usefulness, this paper introduces extensions to address these limitations. The results confirm that the extended BWDM can reduce test case generation time compared to manual methods.

© 2022 *The Author*. Published by Sugisaka Masanori at ALife Robotics Corporation Ltd.
This is an open access article distributed under the CC BY-NC 4.0 license
(http://creativecommons.org/licenses/by-nc/4.0/)

1. Introduction

Software plays an important to role support our daily lives. Hence, its size and complexity have increased, leading to a greater societal impact of software bugs. One significant cause of these bugs is the use of natural language during the early stages of software development, as natural language is inherently ambiguous. To solve this problem, formal methods are adopted in software design. Among these methods, VDM is widely used, and VDM++ serves as a formal specification language for object-oriented modeling within VDM [1].

However, designing software with VDM++ necessitates thorough testing. Manually generating test cases from VDM++ specifications is both time-consuming and labor-intensive, and it may result in incomplete testing. To solve these problems, our laboratory has developed BWDM [2], [3], an automated tool for generating test cases from VDM++ specifications. Despite its usefulness, the current version of BWDM is limited to supporting only integer types and cannot handle test case generation for operations and functions with recursive structures.

To enhance the functionality of BWDM, we have introduced the following extensions:

- A feature to generate test cases for enumerated types
- A feature to generate test cases for operations and functions with recursive structures

This paper is organized as follows. Section 2 details the extended BWDM, Section 3 presents application examples, and Section 4 discusses and validates the effectiveness of BWDM. Finally, Section 5 concludes the paper.

2. The Extended BWDM

This chapter describes on the enhancements in BWDM. Fig. 1 illustrates the structure of the extended BWDM.

2.1. Enhancing Test Case Generation for Enumerated Types

To address the limitation of the original BWDM, which only supported specific types, we enhance both the

Corresponding author's E-mail: takakura@earth.cs.miyazaki-u.ac.jp, kat@cs.miyazaki-u.ac.jp, kita@sun.ac.jp, yamaba@cs.miyazaki-u.ac.jp, aburada@cs.miyazaki-u.ac.jp, oka@cs.miyazaki-u.ac.jp

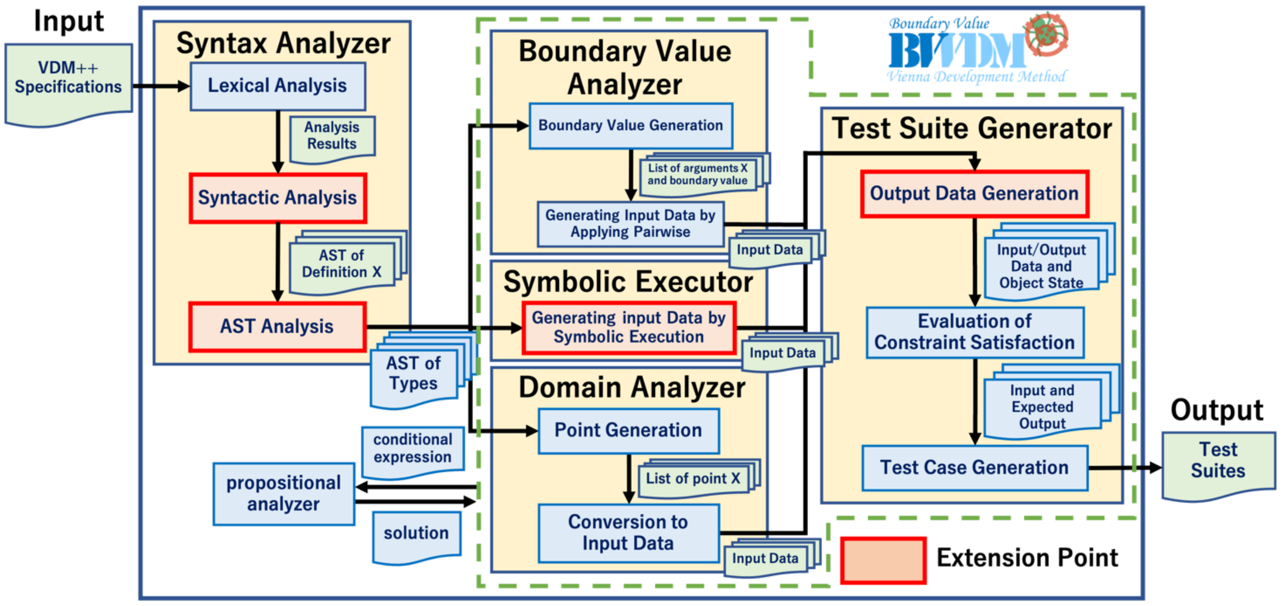

Fig. 1. The structure of the extended BWDM

```
Class judgeLightColor

Types
public TrafficLight = <BLUE> | <YELLOW> | <RED>;
functions
  judgeLightColor: TrafficLight ==> seq of char
    judgeLightCoulor (color) ==
      if color = <BLUE >
        ″The color is blue. ″
      if color = <YELLOW>
        ″The color is yellow″
      if color = <RED>
        ″The color is red.″

end judgeLightColor
```

Fig. 2. VDM++ specification using enumerated type

Syntax Analyzer and Symbolic Executor. Specifically, BWDM is extended to generate test cases for enumerated types, which consist of a set of unique identifiers.

In the updated Syntax Analyzer, if an enumerated type definition is detected during the abstract syntax tree analysis, the type declaration is replaced with a list containing the declared name and values. When this list is found in the syntax tree within the Symbolic Executor, JavaAPI [4] is used to generate enumerated types, creating constraints for the SMT solver [5] used in the propositional analyzer section. This process enables the generation of input data for enumerated types through symbolic execution.

```
Function Name : judgeLightColor
Argument Type : color:
Return Type : seq of (char)

Test Cases by Symbolic Execution
No.1 : <BLUE> -> ″The color is blue″
No.2 : <YELLOW> -> ″The color is yellow″
No.3 : <RED> -> ″The clolor is red″
```

Fig. 3. Output when Fig.2 is applied to the extended BWDM

2.2. Enhancing Test Case Generation for Recursive Operations/Functions

To address the limitation of the original BWDM in generating test cases for operations and functions with recursive structures, we enhance both the Syntax Analyzer and Test Suite Generator.

In the original BWDM, test cases could not be generated if the parsing of a called operation/function was incomplete during test suite generation. This issue prevented the generation of an abstract syntax tree for operations/functions with recursive calls.

To resolve this, we introduce an upper limit on the number of recursive calls for operations or functions with self-recursive calls. The abstract parse tree analysis process is modified to be executed after the parse tree analysis for all definitions is completed. This ensures that test data generation occurs only after the complete parsing of the VDM++ specification. Additionally, a function is added to terminate the process if self-recursive calls exceed the set limit during the output data generation phase in the Test Suite Generator.

3. Application Examples

This chapter presents examples to verify the functionality of the extended BWDM.

3.1. Verification of Test Case Generation for Enumerated Types

Fig. 2 illustrates a VDM++ specification that use an enumerated type for verification purposes, and Fig. 3 displays the results obtained using the extended BWDM. In Fig. 2, the "judgeLightColor" function defines "public TraficLight = <BLUE>|<YELLOW>|<RED>;" as the type. Fig. 3 demonstrates that test cases can be successfully generated for this enumerated type definition.

3.2. Verification of Test Case Generation for Recursive Definitions

Fig. 4 shows a VDM++ specification with a recursive structure used for verification, and Fig. 5 shows the corresponding results using the extended BWDM.
In Fig. 4, the "calcSum" function is defined with a recursive structure. Fig. 5 confirms that test cases can be generated for this recursive definition.

```
Class sumOfNaturalNumbers

  calcSum: nat ==> nat
    factorial (value) ==
      if value = 0 then
        0
      else value + calcSum (value - 1)

end sumOfNaturalNumbers
```

Fig. 4. VDM++ specification with recursive structure

```
Function Name : sumOfNaturalNumbers
Argument Type : value:nat
Return Type : nat

Boundary Values for Each Argument
value: 4294967295 4294967294 0 -1

Test Cases of Boundary Values
No.1 : 4294967295 -> Undefined Action
No.2 : 4294967294 -> Undefined Action
No.3 : 0 -> 0
No.4 : -1 -> Undefined Action
```

Fig. 5. Output when Fig.4 is applied to the extended BWDM

4. Discussion

4.1. Verification of Test Case Generation for Enumerated Types

As discussed in Section 3.1, the extended BWDM successfully generates test cases from definitions using enumerated types in the VDM++ specification shown in Fig. 2. This confirms that the extension allows BWDM to handle enumerated type definitions, thereby enhancing its usefulness.

4.2. Evaluation of Test Case Generation for Recursive Definitions

As detailed in Section 3.2, the extended BWDM is capable of generating test cases from definitions with recursive structures in the VDM++ specification shown in Fig. 4. This demonstrates that the extension enables BWDM to manage recursive definitions, thus improving its effectiveness.

4.3. Comparison of Test Case Generation Time with Manual Effort

We evaluate the time required to generate test cases using the extended BWDM for VDM++ specifications that include enumerated types and recursive structures, comparing it to manual efforts. The VDM++ specifications in Fig. 2 and Fig. 4 are used for this experiment. Manual verification is conducted by five students: two graduate students and three fourth-year undergraduates from our laboratory. The time taken to generate comprehensive test cases without omissions is recorded. The process is halted once the correct test cases are produced, and any errors in the manually generated test cases are noted. The comparison results are presented in Table 1.

Table 1. Comparison of test case generation time

	Time
Average of 10 subjects	14m29s
BWDM	2.6s

Table 1 indicates that using the extended BWDM saves approximately 14 minutes compared to manual test case generation. Additionally, human errors have been observed in the manual process. This study has confirmed that the extended BWDM, which includes functions for generating test cases for enumerated types and recursive structures, reduces both the time and errors associated with manual test case generation. Therefore, the extension of BWDM enhances its usefulness.

4.4. Related Works

Ahmad Mustafa et al. conducted a systematic literature review on automatic test case generation from requirement specifications [6]. They identified and discussed 30 primary studies out of 410, highlighting that most software testing errors from issues in natural language requirements. Detecting ambiguities and incompleteness in natural language is challenging and remains an important problem in requirements.

On the other hand, BWDM uses the formal specification language VDM++ for test case generation, producing test cases from rigorous specifications that eliminate the ambiguities and incompleteness. Therefore, BWDM does not face the issues in test case generation from requirement specifications discussed in [6].

Some approaches to test case generation use UML as the input [7], [8]. However, it is not possible to capture all the different characteristics of a system from UML.

In contrast, BWDM uses the formal specification language VDM++ for test case generation. Hence, it is possible to capture all the different characteristics of a system from a detailed VDM++ specification.

Aamer Nadeem et al. proposed a method for automatic test case generation for VDM++ specifications [9]. This method determines input data using pre-conditions and invariant conditions described in the instance variable definition block. These conditions are equivalence partitioned, and a representative value is randomly selected from the valid input domain. Additionally, a test sequence is generated using a manually prepared test descriptor defined as a valid test sequence. The input data and test sequences are then combined to generate test cases.

In contrast, BWDM can generate test cases solely from a VDM++ specification. Furthermore, test cases generated by BWDM can be used for boundary value testing and domain analysis testing. Test cases generated through symbolic execution can also cover execution flows that boundary value analysis might miss.

5. Conclusion

To enhance the usefulness of BWDM, an automatic test case generation tool for VDM++ specifications, two important extensions have been implemented. These extensions address the limitations of supporting only integer types, and the inability to generate test cases for operations and functions with recursive structures.

The application examples of the enhanced BWDM were presented, demonstrating its ability to generate test cases for definitions using enumerated types and recursive structures, thereby broadening the range of supported types. It has been confirmed that the extended BWDM reduces test case generation time by approximately 14 minutes compared to manual methods. Additionally, the extended BWDM effectively eliminates human errors.

Consequently, the enhancements described in this paper improve the usefulness of BWDM.

Future work includes:

- Extending support to types beyond integers and enumerated types
- Generating test cases for input values with a higher number of recursive calls
- Generating test cases for mutually recursive functions

References

1. Overture Project. Manuals. https://www.overturetool.org/documentation/manuals.html. (Accessed: 2023-12-18).
2. T. Katayama, F. Hirakoba, Y. Kita, H. Yamaba, K. Aburada, and N. Okazaki. Application of Pirwise Tsting into BWDM which is a Test Case Generation tool for the VDM++ Specification. Journal of Robotics, Networking and Artificial Life, Vol.6, No.3, pp.143-147, 2019.
3. T. Muto, T. Katayama, Y. Kita, H. Yamaba, K. Aburada, and N. Okazaki. Expansion of Application Scope and Addition of a Function for Operations into BWDM which is an Automatic Test C55-262, 2022.
4. Z3 Prover/z3. htt ases Generation Tool for VDM++ Specification. Journal of Robotics, Networking and Artificial, Vol.9. No.3, pp.2ps://github.com/Z3Prover/z3. (Accessed: 2024-6-27).

5. Z3: Package com.microsoft.z3. https://z3prover.github.io/api/html/namespacecom_1_1microsoft_1_1z3.html. (Accessed: 2024-6-27).
6. Ahmad Mustafa, Wan M. N. Wan-Kadir, Noraini Ibrahim, Muhammad Arif Shah. Automated Test Case Generation from Requirements: A Systematic Literature Review. Computers, Materials & Continua, vol. 67, no.2, pp. 1819-1833, 2021.
7. M. Lafi, T. Alrawashed, A. M. Hammad. Automated Test Cases Generation From Requirements Specification, International Conference on Information Technology (ICIT), pp. 852-857, 2021.
8. Mauricio Rocha, Adenilso Simão, Thiago Sousa. Model-based test case generation from UML sequence diagrams using extended finite state machines. Software Quality Journal, Volume 29, Issue 3, pp.597-627, 2021.
9. Aamer Nadeem, Muhammad Jaffar-Ur-Rehman. Automated Test Case Generation from IFAD VDM++ Specifications. SEPADS 05: 4th WSEAS International Conference on Software Engineering, Parallel & Distributed Systems, No.28, pp.1-7, 2005.

Authors Introduction

Mr. Shota Takakura

He received the Bachelor's degree in engineering (computer science and systems engineering) from the University of Miyazaki, Japan in 2023. He is currently a Master's student in Graduate School of Engineering at the University of Miyazaki, Japan. His research interests include software testing, software quality, and formal method.

Dr. Tetsuro Katayama

He received a Ph.D. 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 the 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.

Dr. Yoshihiro Kita

He received a Ph.D. degree in systems engineering from the University of Miyazaki, Japan, in 2011. He is currently an Associate Professor with the Faculty of Information Systems, University of Nagasaki, Japan. His research interests include software testing and biometrics authentication.

Dr. Hisaaki Yamaba

He received the B.S. and M.S. degrees in chemical engineering from the Tokyo Institute of Technology, Japan, in 1988 and 1990, respectively, and the Ph D. degree in systems engineering from the 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.

Dr. Kentaro Aburada

He received the B.S., M.S, and Ph.D. 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 networks and security. He is a member of IPSJ and IEICE.

Dr. Naonobu Okazaki

He received his B.S, M.S., and Ph.D. 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.