Black-box approach for software testing based on fat-property

Mengqing TanLi*, *Yan* Jiang, *Xiang* Wang, and *Rushu* Peng

School of Mech. & Eng. University of South China, Hunan, Hengyang, China, 421001

Keywords: Fat-property, Testing case, Black-box software testing, Software engineering.

Abstract. After a useful and summarized procedure of software testing is put forward based software engineering view, this paper proposed a definition of fat-property according to software testing activity in product quality monitoring software. Based on fat-property, black-box testing approach is deeply investigated. In unit testing, equivalence partitioning should include two aspects: data inputting type and function operating type. And a key point of black-box testing is design of boundary / sub- boundary testing case for data inputting type. In integration testing, Sandwich mode should be applied to improve coverage. In validation testing, keynote function and non- keynote function may be tested respectively to accelerate speed of testing and assure coverage of function. System testing based on black-box according to actual usage of software product is very important, and it will determine the quality level of software product.

1 Introduction

Software engineering is the means by which we attempt to produce all of this software in a way that is both cost effective and reliable enough to deserve our trust, and it is the practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them. [1]

Software testing has played a very important role in software engineering, and now gradually becoming accepted as the successor to the various researchers and industrialists. As a keynote aspect of software engineering, software testing includes several phases shown in figure 1 which influence each other. [2]

Black-box testing mainly used in unit phase, integration phase, validation phase, and verification phase.

Testing of graphic user interface is a difficulty to current software product while graphic user interface has become the keynote of new engineering software, and this paper will investigate mainly the testing of graphic user interface using black-box testing method, including equivalence partitioning and design of testing case. [3,4]

* Corresponding author: tlmq-team@163.com

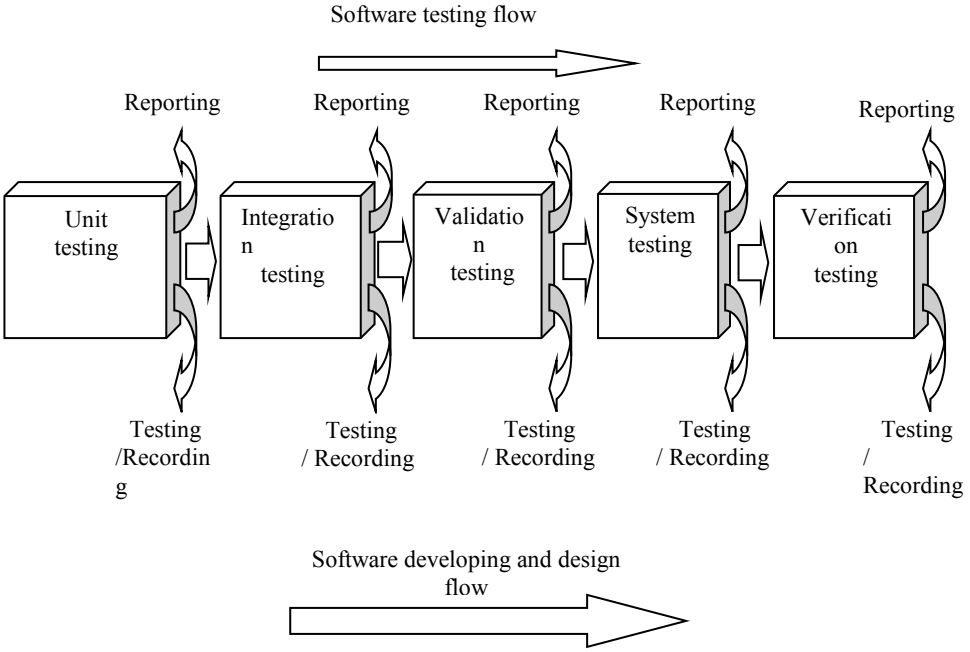

Fig. 1. General procedure of software testing for engineering.

2 Definition and its verification of fat-property [1,2,5]

Fat-property is defined as the property of testing case that volume of testing case is more and more large and will be increasing with testing activity.

Table 1. Fat-property of testing case space for product quality monitoring software.

Unit	Partitioning			Sum of every Unit
	A1-usual effective equivalence kinds	**A2**-Boundary value etc. effective equivalence kinds	**A3**-Ineffective equivalence kinds	
CChart	28	10	45	83
UChart	28	14	44	86
NPChart	28	10	45	83
PChart	28	14	44	86
MedianChart	28	14	44	86
XaveChart	28	14	44	86
XChart	27	10	39	76
Sum	195	86	305	586
Ai/Sum	33.3%	14.7%	52.0%	100%

For the view of software engineering and factual operation, fat-property includes following aspects:

Partitioning. According to equivalence partitioning, effective coverage for fat-property can be reached with scientific evidence.

Representation. Based on representation, the whole testing case space can be covered using less testing cases as possible.

Un-repeating. Testing cases with representation are unrepeated. According to principle of set theory, the element of testing case set has mutual anisotropy property and unordered property, and unrepeated testing cases can be avoided to meaningless testing.

Growing. With the adding of effective testing case, fat-property is growing with the testing activity.

Execution. Similarly, if a testing case is unusable in software testing, this testing case must be deleted. Sequentially, testing cases with representation will be executable and will be actionable factually in testing working.

According to definition of fat-property, we easy to know that fat-property is an absolute attribute and it will be growing with the testing activity. As an illustrated example, table1 has shown the fat-property that all data are derived from the testing case of black-box testing in product quality monitoring software. In this table, the left column is the list of all units in black-box testing, and the last line below has given the proportion of every kind in total partitioning. It is easy to know, volume of usual effective equivalence kinds is 33.3%, but volume of boundary value etc. effective equivalence kinds and ineffective equivalence kinds are 66.7% about 2/3. On the other hand, total volume of testing case will be growing while more and more testing cases are added to improve the effectiveness. Results have indicated that the fat-property of testing case is great striking. [6-10]

3 Black-box testing based on fat-property [1,2]

As you known, Black-box testing based on fat-property is focused on external appearance of software function but internal logical structure, which testing object is taken as a black box.

3.1 Unit testing for black-box testing based on fat-property

3.1.1 Equivalence-partitioning.

In equivalence-partitioning, testing case is usually divided into two parts: effective kinds and ineffective kinds.

For testing of current software product with graphic user interface, equivalence partitioning could be done according to two aspects: data inputting type and function operating type.

For data inputting type, effective kinds should include typical value of reasonable range of inputting data, boundary value and sub-boundary value etc., ineffective kinds should include value beyond range of inputting data, and other abnormal value of combination state.For function operating type, effective kinds should include correct running of functional elements, ineffective kinds should include incorrect operation state and unusual or abnormal combination.

3.1.2 Testing case design [6-8].

According to detailed equivalence-partitioning above, the design of testing case could be done obligated to following four rules: (a) Rule of representation property of testing case, (b)

Rule of unrepeated property of testing case, (c) Rule of decidability property of testing result, (d) Rule of reappeared property of testing result.

Detail disposing for equivalence partitioning of data inputting type should be considered referencing following aspects:(a) For Var∈[Lower, Upper], a typical value can be given as an effective kinds, and boundary value and sub-boundary value are added as addition of effective kinds. Ineffective kinds should be these values beyond [Lower, Upper]. (b) For inputting of all kinds of data type, respective type testing should be design including combination of various ineffective types.

Detail disposing for equivalence partitioning of function operating type should be regarded referencing following aspects:(a) Own effectiveness of running state to every function.(b) Transforming effectiveness between running states.

3.2 Integration testing for black-box testing based on fat-property

Organization of integration testing is very important because of large volume of testing units and their integration mode. A lot of testing practice has verified that Sandwich mode would improve efficiency and coverage of integration testing, shown in figure 2.

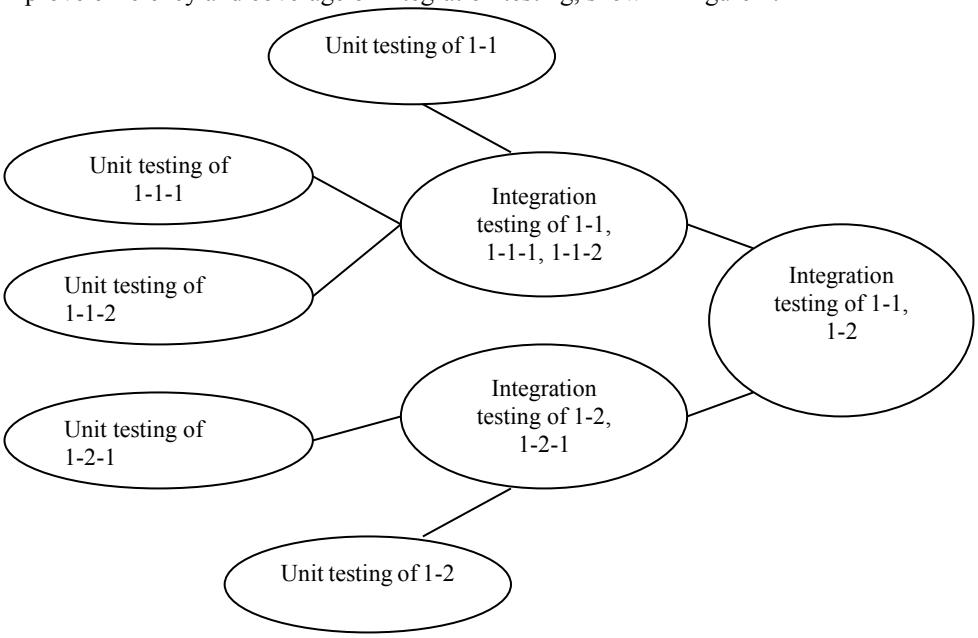

Fig. 2. Sandwich mode of integration testing for same fat-property.

3.2.1 Equivalence-partitioning

Similarly, in two kinds of equivalence-partitioning: effective kinds and ineffective kinds, .for integration testing of current software product with graphic user interface, equivalence partitioning could be done according to two aspects: single functional path operating - state type and multi functional path operating - state type.

In integration testing for single functional path operating - state type, when single functional path and executive state are correct, it should be an effective partitioning.

Otherwise, when single functional path and executive state are incorrect, it should be an ineffective partitioning.

In integration testing for multi functional path operating - state type, if functional path and executive state are all correct, it should be an effective partitioning. But, if any functional path or executive state is incorrect, it should be an ineffective partitioning.

3.2.2 Testing case design [6-8]

Following aspects should be investigated for testing case design in integration tesing: (a) When assembly of units can be correctly running along a reasonable path and arrive at a proper state with accessible transferring parameters, it should be regarded as a testing case of effective partitioning, otherwise as a testing case of ineffective partitioning. (b) If a unit can be addressed as a proper state with no mistakes, it should be taken as a testing case of effective partitioning, otherwise as a testing case of ineffective partitioning. (c) When interface data or parameters are added/updated in source site, and correct results can be displayed or accessed along a reasonable path in target site, it should be regarded as a testing case of effective partitioning, otherwise as a testing case of ineffective partitioning. (d) Empty value, meaningless value and null value of transferring parameters should be regarded as testing case of ineffective partitioning.

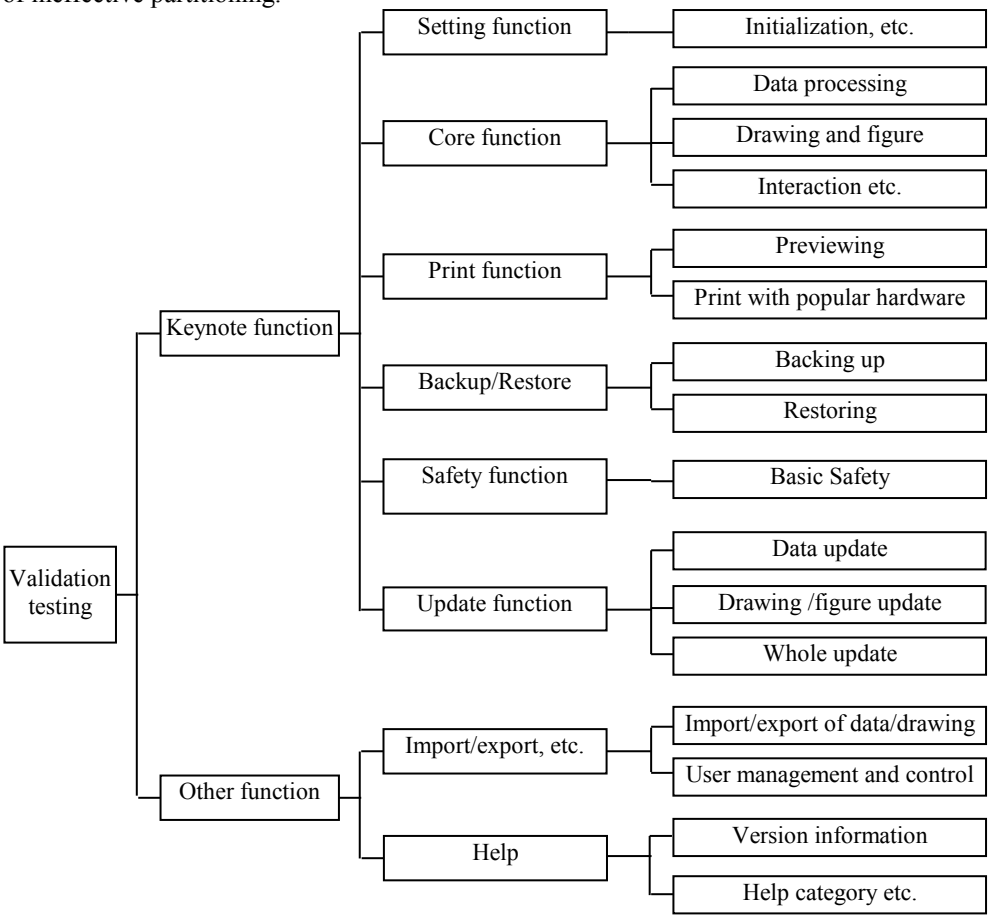

Fig. 3. Main aspects of validation testing based on fat-property.

3.3 Validation testing for black-box testing based on fat-property

Organization of integration testing is very important because of large volume of testing units and their integration mode. A lot of testing practice has verified that Sandwich mode would improve efficiency and coverage of integration testing.

3.3.1 Main aspects of validation testing [6-10]

According to completeness and un-repeat of software testing, main aspects of validation testing for black-box testing based on fat-property is shown in figure 3.

3.3.2 Testing case design

The description of fat-property of testing case in validation testing can divided into a set including three elements as {data mainly using boundary condition, functional path, running state}. And equivalence-partitioning in validation testing is still keep divided mode of effective- partitioning and ineffective-partitioning, but for multi-functional path mode, ineffectiveness of any functional path must be considered for all possibility. Testing case design should be done according to equivalence-partitioning which it should be careful for deleting unnecessary or repeated testing case.

3.4 System testing for black-box testing based on fat-property

In software testing, system testing should be taken as a keynote because it would determine truth feeling of user and quality level of software product. Main aspects of system testing for black-box testing based on fat-property is shown in figure 4. [6-10]

4 Increasing volume and update of testing case

With progression of software testing, the property of testing case will become more and more fatting, because of following operations and disposing: (a) In the previous procedure of testing execution, if some bugs are found, then some testing cases must be added for coverage where these bugs have been appeared. (b) If sub-boundary factor must been considered, testing cases should be added to cover sub-boundary condition. (c) If a new approach is invented and validated, testing cases for using new method should be added. (d) Some bugs are found when regression testing is done, and some testing cases must be added to coverage where these bugs have been checked out. (e) If function of software product is added, testing cases about new function and its state transform must be added too.

5 Conclusion

Software will be new growth pole in twenty-first century and software testing would become update and emerging industry. In software engineering, black-box testing is a main and prior mode for software testing. According to scheme of development and sustainability, design of testing case is always a tired and tired work based on scientific equivalence-partitioning, either for daily working or through whole software testing cycle. And fat-property of testing

space will become great striking with continuously progressing of testing activity, which unnecessary and useless testing case should be deleted.

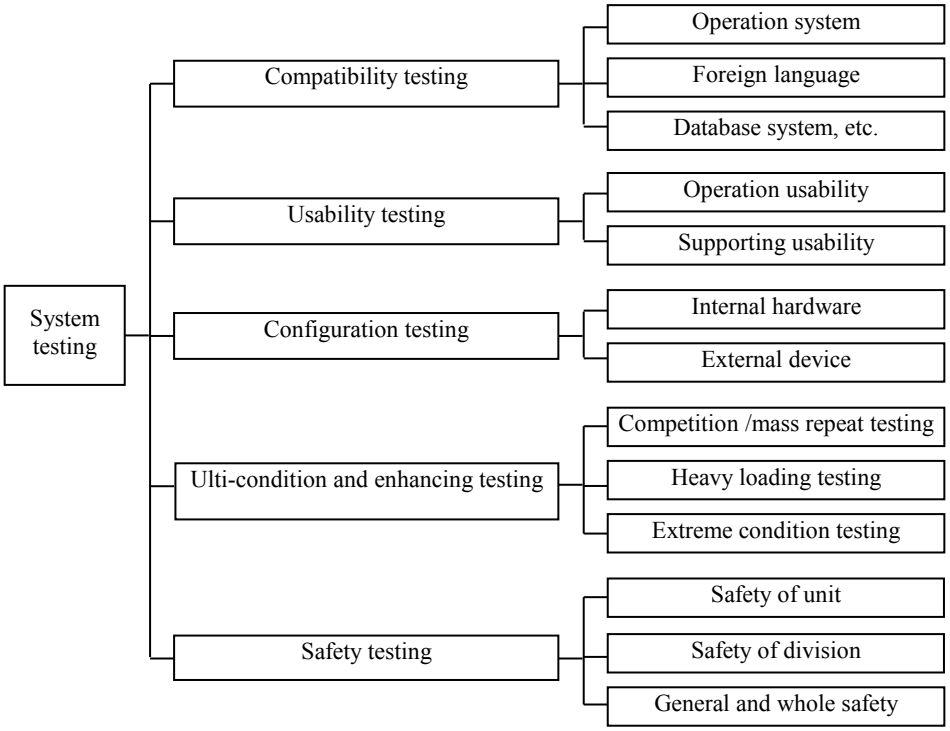

Fig. 4. Main aspects of system testing based on fat-property.

References

1. B. W. Boehm, Classics in software engineering, Yourdon Press Upper Saddle River, NJ, USA, 1979.
2. Per Runeson, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering, 14-2 (2009) 131–164.
3. Pu Zhaoban, Design of Measuring and Control Instrument, China Machine Press, Beijing, 2014 (In Chinese).
4. TanLi Mengqing, Development and simulated test of WEBSPC, IDDME&ACCSPI, Beijing, 2008.
5. Information on http://www.baidu.com, 2019.
6. Mengqing TANLI, Yulin WANG, Xiang WANG, et al., Research on digital inspection of purchasing quality for manufacturing factory, 2nd International Conference on Advances in Management Engineering and Information Technology AMEIT2017, Shanghai, China, 2017, pp. 23-24.
7. MENGQING TANLI, YAN JIANG, YULIN WANG, et al., Digital Inspection of Cutting and Machining Based on Manufacturing Quality for Shop Floor. ICMEIT2018, Shanghai, China, 2018, pp. 1-7.

8. Mengqing TANLI, Yan JIANG, Yulin WANG, et al., Product Quality Monitoring Software Based on Usability for Manufacturing Enterprise, CCME2018, Shanghai, China, 2018, pp. 32-37.
9. TanLi Mengqing, Jiang Yan, Wang Xiang, et al., Research on Quality Tests for Shop Floor Application Based on Local Network, International Conference on Automatic Control and Information Engineering ICACIE2016, Hongkong, China, 2016, pp. 0145-0150.
10. Dan Tang, Mengqing TanLi, Yan Jiang, et al., Product Quality Monitoring of Shewhart Chart Based on Function integration for Manufacturing Factory, 4th Annual International Conference on Information System and Artificial Intelligence ISAI2019, Changsha, Hunan, China, 2019 pp. 123-130.