Grey-Box Technique of Software Integration Testing Based on Message

Mengqing TanLi¹*, Ying Zhang², Yulin Wang² and Yan Jiang²
¹School of Software, University of South China, Hengyang, Hunan, China
²School of Mechanical Engineering, University of South China, Hengyang, China
Email: TLMQ-TEAM@163.COM

Abstract. In the software integration testing, the combined exploding problem was appeared for GUI (Graphic User Interface) software with many advantages. Based on the engineering thinking of staging and the feature of GUI software, the grey technique of integration testing for GUI Software based on message is proposed to solve this problem, in which, the white-box method is used to deal with the consistent and unified part and black-box method is mainly engaged to dispose the difficult and varied part of software testing. By analysis of basic GUI software, the application of this grey technique will improve the testing efficiency with 333% according to testing executed number, and the efficiency in terms of testing executed time will be 438%. An actual example in product quality monitoring software has verified the approach and its efficiency of the grey technique.

1. Introduction and Background
In all phases of software producing, there are “three 40%s”. That is, software testing is occupied with 40% of total workload [1-4], and regression testing has also the ratio of 40% of whole testing working [1-5], while the software defects found in the integration testing is also the 40% of summary software faults and bugs discovered in all testing tasks [1-6]. For today software products, the GUI-Graphic User Interface has already instead of the DOS interface, it is an important innovation in the developing processing of computer software technology, and it has greatly decreased the distance between computer and users with the feature of lower operation difficulty and better user-experience and commercial usability. Otherwise, it has produced a great difficulty for software testing, because the combination exploding problem was appeared in the processing of software testing with these advantages, especially for black-box testing of software. Of course, these trade-offs include many execution paths for a function, data inputting verification with fewer operation mistakes, and default disposing with better efficiency.

2. Base of Integration Testing and Grey-Box Testing
Comparing with the previous programming technology, object-oriented technology has lifted a great pace for software technology, and its advantages and difficulties has changed from units to the interaction and linkage among units. The software testing technology is similar to this changing, the unit testing is relatively easier, and the integration testing is relatively more difficult for general GUI software/system. In particular, it happened when most of unit testing must be finished by programmer self and only main and key unit should be tested by software tester for the arrangement of testing department. And this state is appearing more frequently for the integration testing of update GUI software.
Generally, there are two modes of testing strategy in integration testing of GUI software including “big exploding mode” and “incremental mode”. The former is seldom used today except very small software program, and another is widely applied in integration testing of GUI software. The “incremental mode” has several types, but the “improved sandwich mode” has more efficiency for update integration testing of GUI software which every unit is able to be tested independently distinguished with “sandwich mode” [7]. The figure 1 has illustrated the “improved sandwich mode”.

![Figure 1](image1.png)

**Figure 1.** The integration testing strategy of “improved sandwich mode”.

Integration testing of object-oriented software has been the keynote and difficulty point for the developing of software testing. Many scholars and industrial practitioners have engaged a lot of research working to solve integration testing of GUI software. Alessandro Orso has done systematic research for the strategy and method of integration testing [6], and Emil Al’egroth, Robert Feldt have taken a deeply case study for sustained application of GUI testing techniques with analysis of advantages and disadvantages [8].

The programming method of today software product is mainly based on object-oriented visualization technology with GUI interface, in which, the key mechanism of the object-oriented visualization technology is the message map based on event. This mechanism has simplified the programming logic and improved the communication mode, and integration testing technique of object-oriented software with GUI interface can use this mechanism flexibly, detail will be discussed in section 3.1.

The grey-box testing is a “translucent” testing approach, and a synthetic testing method with white-box testing and black-box testing. What a nice coincidence, the primary application of the grey-box testing is that of integration testing, including testing of database product [6]. Figure 2 has illustrated the concept and definition of grey-box testing, and table 1 has analyzed the deference among white-box testing, black-box testing and grey-box testing [9].

![Figure 2](image2.png)

**Figure 2.** The concept and definition of grey-box testing.
Table 1. The deference among white-box testing, black-box testing and grey-box testing.

|                               | White-box testing | Black-box testing | Grey-box testing |
|-------------------------------|-------------------|-------------------|-----------------|
| **Main applied phase**        | Unit testing      | Integration/validation testing, system testing | Integration/validation testing, system testing |
| **Testing derivation type**   | Programming-based mainly | Requirement specification and design specification - based | Programming-based, Requirement specification and design specification –based |
| **Static and/or dynamic testing** | Static and dynamic | Static mainly |
| **Requirement of tool**        | High              | Higher            | Lower           |
| **Requirement of personnel**   | High              | Lower             | Higher           |
| **Advantage, disadvantage and application** | Easy to find: hidden code error. Disadvantages: duty is unclear between developer and tester, being unfitted to network and distributed system, and testing tool is complex. | Easy to find following error: function error, interface error, data structure and DB visiting error, performance fault, and initialization error. | Balancing between white-box and black-box testing, giving clear duty, easy to disposing flexibly, and specially for data type, communication protocol, and exception handling. |

3. Grey-Box Technique of Integration Testing for GUI Software

Using the basic thinking of “staging” of software engineering and starting from the feature of GUI software, the grey technique of integration testing for GUI Software based on message is proposed to solve the combination exploding problem.

3.1. Principle of Grey-Box Technique in Integration Testing

The grey technique of integration testing for GUI Software is an approach that white-box testing and black-box testing are reasonably synthesized with front white-box testing for message handling mechanism. It will bring into play with the advantages of white-box testing and black-box, and can find the faults and failures as early as possible using the mechanism of message map accompanying with GUI interface, and can consequently improve the testing efficiency [10]. The principle of grey technique of integration testing for GUI Software is shown in figure 3.

In figure 3, after testing the correctness from front units, that is interface unit 1, ……, interface unit n, to the entrance of map function, we can only test the message path \([r_1 \rightarrow r_2], \{r_1 \rightarrow r_3\}, \{r_1 \rightarrow \ldots\}, \{r_1 \rightarrow r_e\}\), it will cover all message path \(\{r_2 \rightarrow r_3\}, \{r_1 \rightarrow r_3\}, \{r_1 \rightarrow \ldots\}\), \(\{r_2 \rightarrow r_3\}\), \(\{r_2 \rightarrow r_e\}\), \(\{r_1 \rightarrow r_3\}\), \(\{r_1 \rightarrow r_e\}\), \(\{r_1 \rightarrow \ldots\}\) with this disposing approach, the testing efficiency will be improved greatly. In factual operation procedure, the information dialogue indicating the medium processing state can be used, for example, the function MessageBox () can be used to output testing information.

The selection of pole position should be taken note of: (1) the path aggregation point across white-box analysis; (2) the point that it must be go through; (3) the entrance point of map function; or (4) the entrance point of initial member function.
Advantages of this grey technique of integration testing for GUI Software are: (1) testing and review working is not very complicated before the instrumentation position, it is convenient to control the testing workload; (2) the running mechanism before the instrumentation place is generally simple, and the testing path is universal; (3) it is easy to find the error by white-box testing before the entrance point of map function or initial member function because of the carelessness of programmer.

3.2. Efficiency Analysis of Grey-Box Technique

As above, application of grey-box technique is to improve the efficiency of software testing, consequently, following is the concrete analysis according to testing executed number and testing executed time.

3.2.1. Computation of Testing Executed Number and Efficiency Analysis. Now, let \( f \) take as the executed number of front events, and suppose that \( e \) is the executed number of map functions. So, the efficiency according to testing executed number is:

\[
c = \frac{e \cdot f}{e + f} \times 100\%
\]

Generally, front events have: Menu Item, Toolbar, Pop Menu Item and Hot Key etc. we can let \( f = 4 \), and in terms of the advanced feature of GUI software, let \( e = 20 \). So

\[
c = \frac{e \cdot f}{e + f} \times 100\% = \frac{20 \times 4}{20 + 4} \times 100\% = 333\%
\]

3.2.2. Computation of Testing Executed Time and Efficiency Analysis. Similarly, let \( t_1 \) take as the executed time of front event, and suppose that \( t_2 \) is the executed time of map functions. So, the efficiency according to testing executed time is:

\[
c = \frac{(t_1 + t_2) \cdot e \cdot f}{t_1 + (t_2 \cdot e)} \times 100\%
\]

Here, the executed time of front event is calculated in average, generally let \( t_1 = 0.1 \) min, and let \( t_2 = 1 \) min. If \( f = 4 \) and \( e = 20 \), we have
In summary, according to above 3.2.1 and 3.2.2, we have a conclusion: in integration testing for GUI Software, the application of grey-box technique based on message path will greatly improve the testing efficiency with volume of 333% in terms of testing executed number, and gain more improvement about 438% according to testing executed time.

4. Example
Product quality monitoring software - PQMS2.0 is a kind of engineering software based on Windows interface [7, 11] and it has used visual programming technology. In PQMS2.0, the visual interface includes many popular interface units - Menu Item, Toolbar, Pop Menu Item and Hot Key etc. and the function operating paths embodies convenience and usability with many choices. Based on this state, the integration testing of “adding inspection data and print quality control chart” in PQMS2.0 is used as an actual example for verifying the former approach of grey-box technique for GUI software.

In order to assure the testing coverage, the above testing strategy of “improved sandwich mode” can be applied in PQMS2.0 as shown in figure 4. For improving the integration testing efficiency based on product baseline, the “Smoking testing” [12] may be firstly executed before total regression testing [4, 5, 13, 14].

\[
c = \frac{(t_1 + t_2) \cdot e \cdot f}{t_1 + (t_2 \cdot e)} \times 100\% = \frac{(0.1+1) \times 20 \times 4}{0.1+1} \times 100\% = 438\%
\]

In PQMS2.0, a lot of message path can finish the function of “adding inspection data and print quality control chart” along with all kinds of event triggers. The integration testing of “adding inspection data and print quality control chart” for upgrade version, including smoking testing, can be applied grey-box approach discussed above according to message path.

According to figure 3, in certification of grey-box technique for integration testing of “adding inspection data and print quality control chart” of PQMS2.0, interface units include (a) Pop Menu Item – “Monitoring Category-Adding-Quality inspection data”, (b) Menu Item – “Quality monitoring data-Sheet mode”, (c) Toolbar – “Inspection data” and (d) Hot Key – “‘Alt+I’- ‘S’”. Message handling is given in “BEGIN_MESSAGE_MAP … END_MESSAGE_MAP ()”. Map function is “BOOL CInspectionDataDIALOG::OnInitDialog()”. Handling units include (a) Adding inspection data manually, (b) Import inspection data from digital measuring gauge, (c) Calling inspection data from saving, and (d) Adding inspection data using the Notepad of Windows.

By actual operation of testing, we get executed time of front events as following: 0.17 min, 0.03 min, 0.03 min and 0.13min. And the testing executed time of applying grey-box technique in example is list in table 2. And in table 2, P0 is the start point, and P1 is the entrance point of map function or

![Figure 4. The integration testing strategy in example.](image-url)
initial member function, and P2, P3, P4 imply respectively the point of data saving, data displaying, and control chart displaying and print.

From table 2, we can know that the sum of testing executed time is 34.16 min for applying grey-box technique in integration testing of “adding inspection data and print quality control chart”.

Table 2. The executed time\(^a\) of applying grey-box technique in example\(^b\)/min.

| Path   | P0 | P1 | P2 | P3 | P4 | P5 | Sum | Note        |
|--------|----|----|----|----|----|----|-----|------------|
| Path rI | 0.17 | 5.25 | 0.15 | 1.75 | 7.32 |     |     | Adding data manually, 20 data |
| Path rII | 0.17 | 5.50 | 0.20 | 4.63 | 10.50 |     |     | Importing data from digital measurement tool, 12 data |
| Path rIII | 0.17 | 4.50 | 0.33 | 3.08 | 8.08 |     |     | Calling data from saving, 60 data |
| Path rIV | 0.17 | 2.33 | 0.33 | 5.43 | 8.26 |     |     | Importing data with the Notepad of Windows, 40 data |
| Sum     | 0.68 | 17.58 | 1.01 | 14.89 | 34.16 |     |     |                         |

\(^a\) Volume testing and preparing time is not considered.

\(^b\) The number of testing data is considered in general situation.

Otherwise, testing executed time under previous testing mode of testing all message path can computed as following:

- Testing path I—Pop Menu Item.
  \(M_1 = 34.16\) min
- Testing path II—Menu Item.
  \(M_2 = 4 \times 0.03 + (17.58 + 1.01 + 14.89) = 33.6\) min
- Testing path III—Toolbar.
  \(M_3 = 4 \times 0.03 + (17.58 + 1.01 + 14.89) = 33.6\) min
- Testing path IV—Hot Key.
  \(M_4 = 4 \times 0.13 + (17.58 + 1.01 + 14.89) = 34\) min

Summing,
\[ M = M_1 + M_2 + M_3 + M_4 = 135.36\) min

So, the testing executed efficiency for using grey-box technique in example is:
\[ c = \frac{135.36}{34.16} \times 100\% = 396\% \]

By above computation, we can know that the efficiency is accelerated about 400% for integration testing of “adding inspection data and print quality control chart” in PQMS2. That is, one tester could do 4 times workload applying grey-box technique based on message than previous method of testing all message paths.

5. Discussion and Conclusion

Further, from above equations, we can get some results. For the state of more executing paths with more front events, the efficiency of applying grey-box technique based on message will be greatly improved. Additionally, for more long time and more complex operation procedure after the entrance point of map function or initial member function, the effect of applying grey-box technique will also be remarkable. In summary, for update GUI software, the grey-box technique based on message in integration testing is a fitted approach to solve the combination exploding problem. In fact, the white-box method is used to deal with the consistent and unified part and black-box method is mainly engaged to dispose the difficult and varied part of software testing in this grey-box technique. Of course, this approach is aimed at the improvement of testing efficiency. This research by an actual example indicates that the efficiency using this grey-box testing will improve about 400% in terms of testing executed time. This approach can be effectively applied in integration testing work especially for GUI software.
References

[1] Boehm B W 1979 *Classics in Software Engineering* (New Jersey: Yourdon Press Upper Saddle River).

[2] Runeson P 2009 Guidelines for conducting and reporting case study research in software engineering *Empirical Software Engineering* **14** (2) 131-164.

[3] Bradbury J S, Cordy J R and Dingel J 2005 An empirical framework for comparing effectiveness of testing and property-based formal analysis *Proc. Int. of the ACM SIGPLAN-SIGSOFT Work on Program Analysis for Software Tools and Engineering* (Portugal: Lisbon) pp 160-170.

[4] Ma X 2008 *Researching on Regression Testing Selection Technique for Object-oriented software — Test Suite Minimization* (Hangzhou: College of Computer Science, ZheJiang University) chapter 1.

[5] Tang L 2005 *Software Testing Management Technology* (Changsha: Institute of Computer Science, National University of Defense Technology) chapter 1.

[6] Orso A 2001 *Integration Testing of Object-Oriented Software* (Politecnico Di Milano) chapter 3.

[7] TanLi M, Jiang Y and Wang X 2019 Black-box approach for software testing based on fat-property *Proc. Int. Conf. on Computer Science, Communication and Network Security* (Sanya) vol 1 (DEStech Publications, Inc.) pp 198-204.

[8] Al’egroth E and Feldt R 2017 On the long-term use of visual GUI testing in industrial practice: A case study *Empirical Software Engineering* **22** 2937-2971.

[9] Acharya S and Pandya V 2008 Bridge between black box and white box-gray box testing technique *International Journal of Electronics and Computer Science Engineering* **2** (1) 175-185.

[10] Xu Y 2015 A study of test case reuse based on CBR *Software* **36** (9) 117-120.

[11] Tang D, TanLi M and Jiang Y 2019 Product quality monitoring of shewhart chart based on function integration for manufacturing factory *Proc. the 4th Annual Int. Conf. on Information System and Artificial Intelligence* (Changsha) pp 123-130.

[12] McConnell S 1996 Best practices: Daily build and smoke test *IEEE Software* **13** (4) 143-144.

[13] Patton R 2006 *Software Testing* (USA: Pearson Education Inc.).

[14] Fu B 2014 *Course of Software Testing* (China: Tsinghua University Press).