Scientific Research Publishing

Journal of Software Engineering and Applications, 2023, 16, 361-396
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Strategy and Methodology of Integration Testing for GUI Software

Mengqing TanLi[1], Jiyi Xiao[1], Ying Zhang[2]

[1]School of Software, University of South China, Hengyang, China
[2]School of Mechanical Engineering, University of South China, Hengyang, China
Email: TLMQ-TEAM@163.com

How to cite this paper: TanLi, M.Q., Xiao, J.Y., Zhang, Y. (2023) Strategy and Methodology of Integration Testing for GUI Software. *Journal of Software Engineering and Applications*, **16**, 361-396.
https://doi.org/10.4236/jsea.2023.168019

Received: July 1, 2023
Accepted: August 26, 2023
Published: August 29, 2023

Abstract

In this paper, by means of effective testing practices, main strategies of integration testing for GUI software, including differentiating strategy for distinguished system, strategy of personnel organization, incremental testing strategy based on baseline version, testing strategy of circulating loop through the whole life, and the strategy of test suite construction, were briefly investigated. Moreover, for the code analysis, the FTA (Fault Tree analysis) is proposed to deal with the software change in regression testing. For test suite constructing, the constructing methods for baseline version and the incremental change are deeply discussed, in which main points focus on the testing strategy based on "Sheet/Form", the "Grey-box approach" for integration testing process, and the application of the improved STD (State Transform Diagram) in state testing. At the same time, the suite construction of integration testing for two types, including small scale program and large scale software, is analyzed and discussed in detail. For testing execution, the specific method based on "Cross-testing" is investigated. Concurrently, by a lot of examples, all results of testing activity indicate that these strategies and methods are useful and fitted to integration testing for GUI software.

Keywords

Integration Testing, Strategy and Methodology, Grey-Box Approach, GUI Software

1. Introduction and Background

On the high nature and view of study philosophy, we affirm advanced aspect of structural-functionalism, an expressing form of system theory, but do not negate usefulness and value of empiricism with a bit of locality feature sometimes

[1]-[5]. In software testing, we must pay great attention to the analysis of software structure and functionality, e.g. the application of FTA (Fault Tree Analysis) for test suite construction in regression testing [6]. At the same time, summarizing and finding the law and principle of software testing is a very important work in software testing practice, e.g. the strategy of "Grey-box approach" in integration testing [7].

In detail, for software system, on the one hand, it consists of various units including initializing unit, setting unit, executing unit, output unit and help unit etc., on the other hand, the relationship among units generally includes two types, *i.e.* the function addressing and the data connecting. As a consequence, the integration testing of software system must synchronically test and verify both function addressing and data connecting [5].

Additionally, on our opinion, study on the strategy of software testing should be considered in terms of two aspects including organization and technology. In the organization aspect, the personal organizing and process arrangement should be concerned [2]. For the technology aspect, we should notice testing strategy, testing method and approach, testing tool, etc. Moreover, in software testing activity, the graph tool is emphasized here because graph tool will improve the software testing work on quality and efficiency, and usual graph tools are shown in **Figure 1**.

Without deniable evidence, Chinese software development has been put in a great pace, not only on basic computer structure building but also on update communication technology especially mobile communication technology. However, in software engineering, the core basis of software programming and software testing still falls behind, such as programming language and tools and automatic software testing tools. For software testing, we still lack the advanced technology and methodology of software testing all the time, such as for GUI software testing [4]. In this study, we will deeply investigate the strategy and methodology of GUI software testing mainly focusing on integration testing.

2. Related Literature and Work

Ron Patton in his writings "software testing" [3] proposed two ways of integration

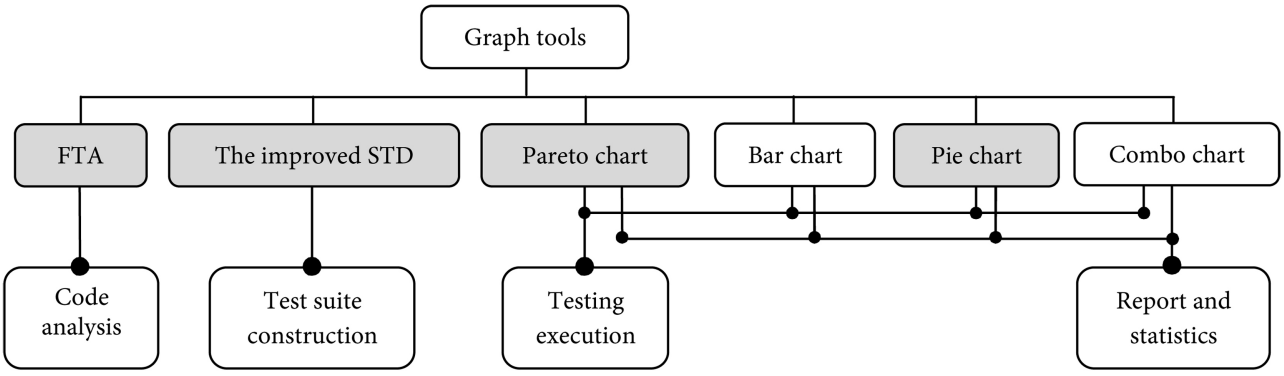

Figure 1. Graph tools of software testing in our research.

testing including bottom-up way and top-down way. Moreover, in his opinion, the driven module can be used to assure testing more completely, and the application of stub will improve the testing velocity. Fu Bing [4] thought that the "Modified Sandwich Integration Testing" has better performance e. g. high parallelism and being easy to execute special path testing. Li Fan [5] gave the suggestion of synthetically applying various method of integration testing in terms of actual situation of software testing activity.

For integration testing of GUI software, Fu Bing [4] has put forwards the difficulty of GUI software testing, and considered that the study of GUI testing, including that of GUI integration testing, still located in the initial phase now and existed testing technology cannot assure the quality of factual GUI software.

So, against the difficulty of the problem of GUI software testing, by summarizing from home and abroad [8] [9] especially from our software testing practice, we proposed that the "Triple-step method" based on "Sheet/Form" to deal with the problem of unit testing of GUI software [2] which details will be discussed in this paper, and the "Grey-box approach" to dispose the problem of integration testing of GUI software [7] which specific examples will be demonstrated in the following.

Recent studies [2] [10] has deeply investigated the unit testing for new testing organization—"Pair-wise" mode, and a previous work [11] has proposed the method of test suite construction for smoke test which can be taken as a special integration testing. Consequently, this study will discuss the strategy and methodology of integration testing for GUI software including code analyzing, test design of baseline version and increment change, testing execution for "cross-testing", etc. As the key aspect, the test suite construction would be depicted with more concerning.

3. Strategy of Integration Testing for GUI Software

As we all known, types and applied areas are distinguished for factual software system. In fact, integration testing may be relative to not only software self but also both software and hardware [12]; at the same time, it may refer to not only functionality but also data and information [11]; even more it may not only consider running in local machine but also consider linking to computer network [12]. All these features require adoption of differentiating strategy.

3.1. Differentiating Strategy for Distinguished System in Integration Testing

As mentioned above, the factual software and system are various. Some software have less functions and less data interfaces, and integration testing may be not complex such as small-scale embedding software and instrument inspection software. In contrast, others have more functions and more data communication interfaces, and integration testing will be complicated and the workload of integration testing will be increased. Without loss of unification, factual software

system can be systematically divided into three types including safety-critical system, general system, and lower system. Consequently, integration has three types, *i.e.* key integration, important integration, and unimportant integration. Thus, execution of integration testing should be respectively done according to these actual types. Furthermore, the arrangement of integration testing must be executed in terms of actual state of software integration, *i.e.* key and important integration should be arranged with more rigid testing while fewer testing items should be briefly tackled to integration for lower system and unimportant parts. The specific strategy of integration testing arrangement is shown in Table 1.

3.2. Strategy of Personnel Organization in Integration Testing

The strategy of personnel organization takes an important role in software testing activity. The new organization embedding "Pair-wise" mode [2] is an updating testing organization that it can keep effective cooperation between programmer and tester. The composition of "Pair-wise" organization mode is shown in Figure 2. In this mode, the testing activity mainly include two parts, *i.e.* cross-testing and independent testing. For cross-testing, the code or program of programmer must be tested by crossing tester under effective monitoring of manger. For independent-testing, key testing including key sampling testing must be done by independent-tester.

3.3. Strategies in Testing Procedure in Integration Testing

The strategy in testing procedure should be "Taking the software testing activity of baseline version as basic center and keeping testing activity effective and rapid responding all through the whole software producing process."

Table 1. The choice of testing items in integration testing for deferent software/system.

Testing items		Safety-critical system			General system			Lower system	
		KI[a]	II[b]	UI[c]	KI	II	UI	II	UI
Code review	Desk check, etc.	○[d]	○	○	○	○	○	○	Δ[e]
Data testing	Process boundary	○	○	○	○	○	Δ	○	[f]
	Format & interface	○	○	○	○	○	Δ	Δ	Δ
	Safety	○	○	○	○	○	○	○	Δ
Functin and state testing	0-switch requirement	○	○	○	○	○	Δ	Δ	
	N-switch requirement	○	Δ	Δ	Δ				

a. KI-Key integration, b. II-Important integration, and c. UI-Unimportant integration, d. "○" presents an item that it must be done, e. "△" implies an alternative item, and f. the blanket is an item not required to do.

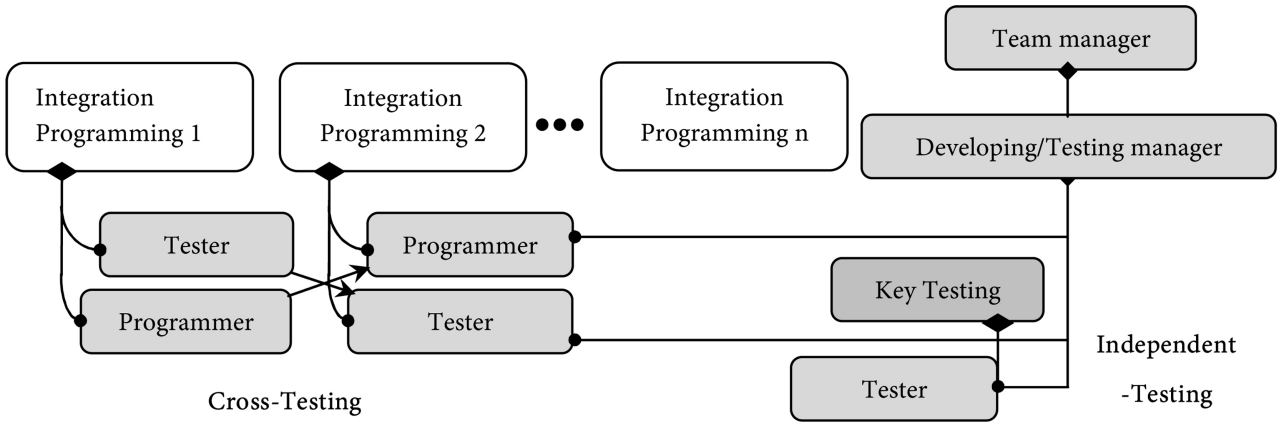

Figure 2. Test organization of integration testing embedding "Pair-wise" mode.

3.3.1. Incremental Testing Strategy Based on Baseline Version

For GUI software, especially for today software with online version update, the general strategy is the incremental testing strategy based on baseline version. With the view of integration testing, this incremental testing strategy can be described as shown in **Figure 3**, in which the smoke test is taken as a brief function testing to determine whether next testing phase is continuously executed.

3.3.2. Testing Strategy of Circulating Loop through the Whole Life

For today's GUI software, because the rapid speed of updating and the mutual close relation among all phase of software producing, testing strategy of circulating loop through the whole life should be adopted rather than dividing testing phase from programming phase, as shown in **Figure 4**. In this strategy, the results of testing phase such as unit testing, integration testing, system testing, and validation testing must back forward to coding and programming phase. In fact, the results of testing phase should back forward to design phase including primary design and detail design, even more to requirement analysis phase.

3.4. Strategy of Test Suite Construction in Integration Testing

In software testing activity, the workload of test design has the ratio of 60% in the whole testing work, and there is an upward trend, especially for the construction of test suite in update software with rapid upgrading [13]. In general, test suite construction should concern many aspects including the building of basic standard, application of advanced strategy and methodology, the set-up of testing environment, and use of history experience [14], etc. and the procedure and strategies of test suite construction can be demonstrated as **Figure 5**. Additionally, if the test design is very difficult, this task can be assigned to another skilled testing engineer or the independent tester.

4. Methodology

For integration testing of GUI software, the factual software system must be firstly delicate to do analysis including using effective method and advanced

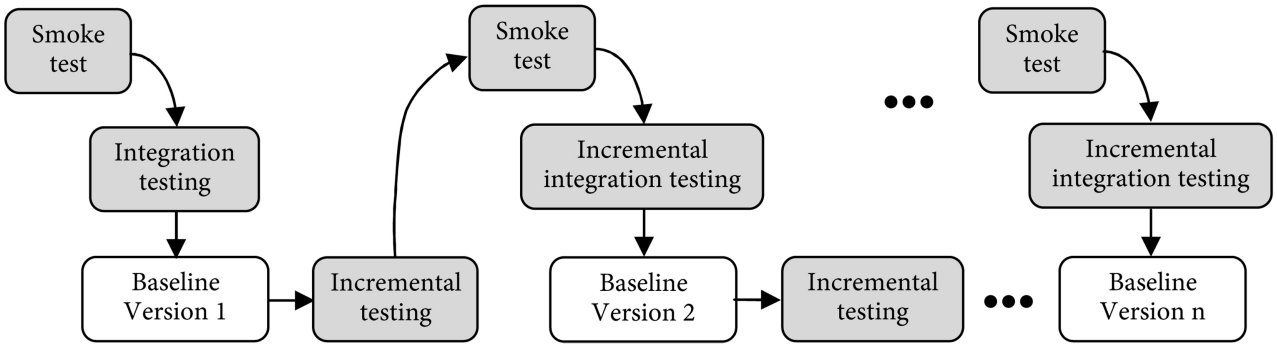

Figure 3. Procedure of incremental testing based on baseline version.

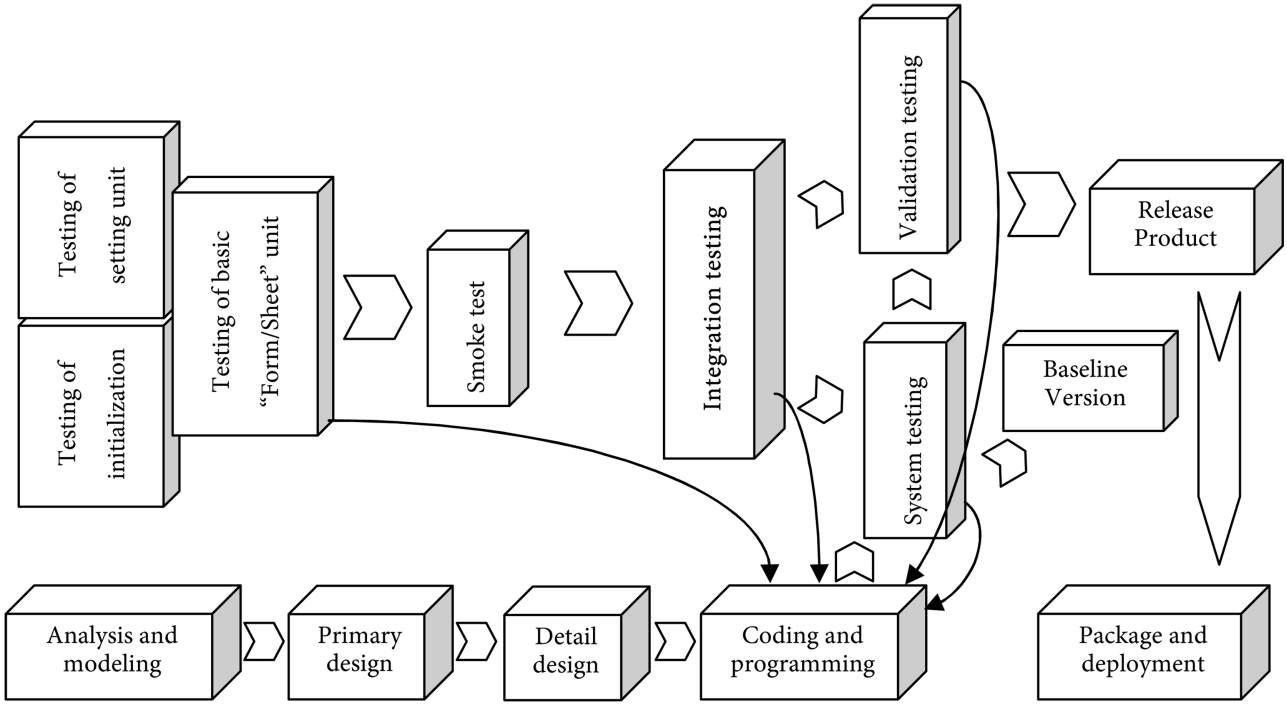

Figure 4. Testing strategy of circulating loop through the whole life.

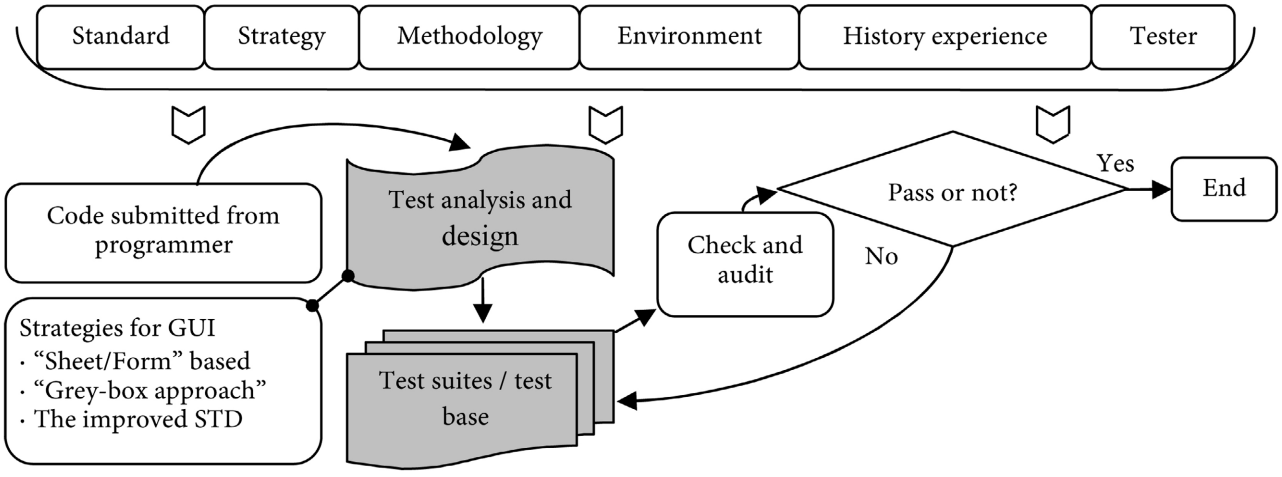

Figure 5. Procedure and strategies of test suite construction in integration testing.

tool. At the same time, in order to assure finding out more BUGs and improve the testing efficiency, the test suite constructing must be paid more attention according to actual requirement, in which the good planning and useful tool are necessary [15].

4.1. Case Software

PQMS2, Product Quality Monitoring System version 2.0, is a control tool of product quality for various manufacturing factory and GUI is shown in **Figure 6**. Using PQMS2, seven kinds of control chart can be drawn, and product quality of divisions of a factory can be monitored, including material and standard part from purchasing, parts and components in producing, and finished product after assembly [16].

In a word, PQMS2 is a factual craft of GUI software. Additionally, with a larger scale, PQMS2 is the application software for Microsoft Windows with more general GUI controls and components, and its integration testing has representativeness and typicality for GUI software.

4.2. Research Design

For software testing, integration testing has some common features, but it also embraces its own distinguished aspects. Generally, the process of integration testing for GUI software has five steps as follows.

Step 1 Analysis of code and program.

Because integration testing starts its activity after all related unit testing are finished, the integration order and method must be considered with systematic view [17]. In general, the modified Sandwich strategy and testing method based

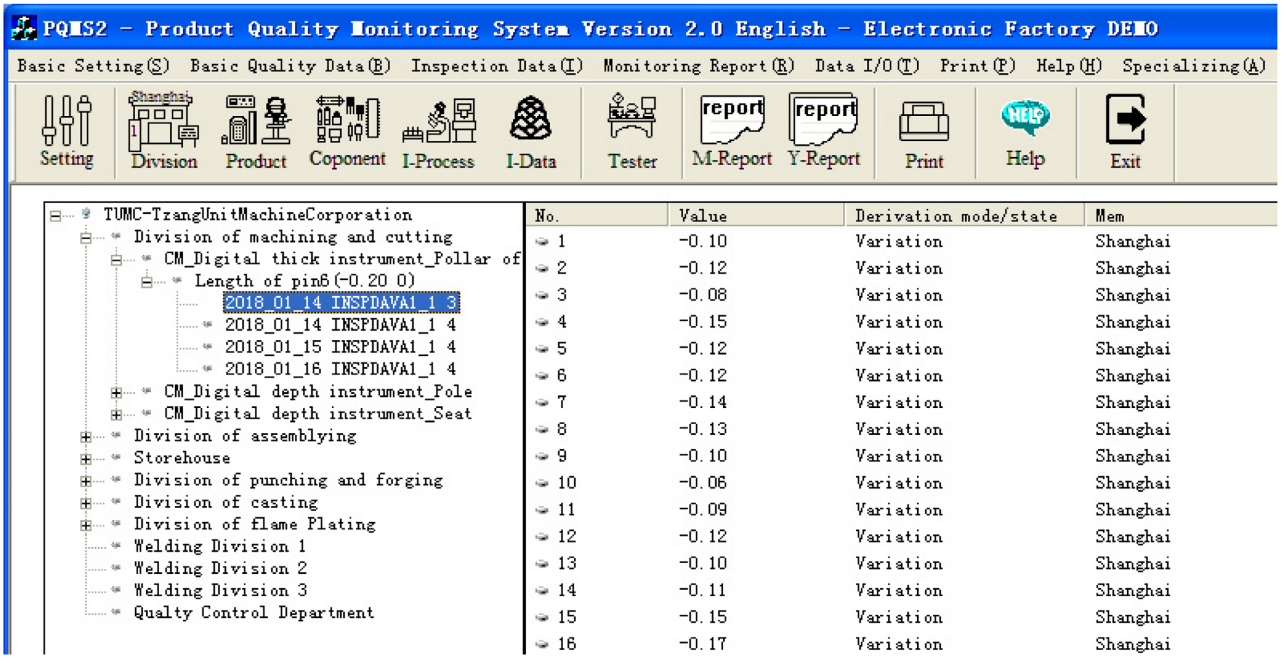

Figure 6. GUI of PQMS2.

on "Sheet/Form" for GUI software should adopted for baseline version. For the incremental testing, the FTA method should be applied for dependency analysis of software change.

Generally, the procedure of integration testing should be also executed in terms of the "Triple-step method", which the data testing is prior to be executed without ignoring process boundary testing, and next is function testing and state testing in sequence. However, it is noticed that some steps can be omitted for factual software. In terms of requirement of Section 3.1, PQMS2 taken as a general GUI system, the specific strategy of testing item assignment is shown in Table 2.

In coding and programming of GUI software, there are generally two situations including the type based on "Sheet/Form" and the type driven directly by member function without "Sheet/Form". The former is the main situation, and its functions are implemented by all controls and components in the "Sheet/Form". For another type, its functions are directly driven by controls and components or hotkey in the window interface.

By a lot of testing practice, we conclude that the unit testing based on "Sheet/Form" is an effective method for GUI software testing, *i.e.* all controls and components in the "Sheet/Form" should taken as a unified entity including these handling by member functions if the function is implemented within the "Sheet/Form". For functions of software are directly driven by controls and components or hotkey in the window interface, the testing should be similarly done in terms of the member function driven by event of the control in the "Sheet/Form", *i.e.* the testing of this situation should be disposed in the level of member function.

As a typical example of GUI software, testing methods in PQMS2 are assigned according to above results, and details in terms of Figure 4 are shown in Table 3.

Table 2. The executed strategy of general testing items in integration testing in PQMS2.

Testing items		Key integration	Important integration	Unimportant integration
Code review	Desk check, etc.	Obliged	Obliged	Obliged
Data testing	Process boundary	Obliged	Obliged	Needed
	Format & interface	Obliged	Obliged	Needed
	Safety	Obliged	Obliged	Obliged
Functin and state testing	0-switch requirement	Obliged	Obliged	Needed
	N-switch requirement	Actual situation	Actual situation	Actual situation

Table 3. The assignment of testing method for preparation of integration testing in PQMS2.

Testing units		"Sheet/Form" based	Member function based	Other method
Unit testing of windows controls and components				•
Unit testing of setting units		•		
Unit testing of initialization units			•	
Unit testing of basic sheets	Unit testing of factory sheet	•		
	Unit testing of division and department sheet	•		
	Unit testing of product or class sheet	•		
	Unit testing of part/component sheet	•		
	Unit testing of inspection process sheet	•		
	Unit testing of inspection data sheet	•		
Unit testing of other units		•	•	•

For update software product, the integration testing of baseline version is the most important node for software testing through the whole life cycle. Based on the basic procedure in **Figure 5**, test suite construction of integration testing of PQMS2 for baseline version can be scheduled in terms of the modified Sandwich method, and the specific scheme is shown in **Figure 7**.

Step 2 Construction of test suite.

For the integration testing of a factual software system, we must consider the factual situation of this system as mentioned in Section 3.1, which may mainly include system scale, software composition, running environment, and application scenario, etc. [18] [19].

According to the result of software testing practice, we know that test design is the central task with about workload of 60% in software testing activity, which the majority is the test suite construction. In integration testing of GUI software, we have proposed three measures to dispose test suite construction as follows.

- Testing method based on "Sheet/Form" for GUI software testing.
- "Grey-box approach" for integration testing.
- Improved STD method for state testing.

For GUI software, a lot of testing work is the work of high-level testing for the general software system besides traditional testing of logic testing and program verification. In order to be easily understood and to demonstrate it, the work of

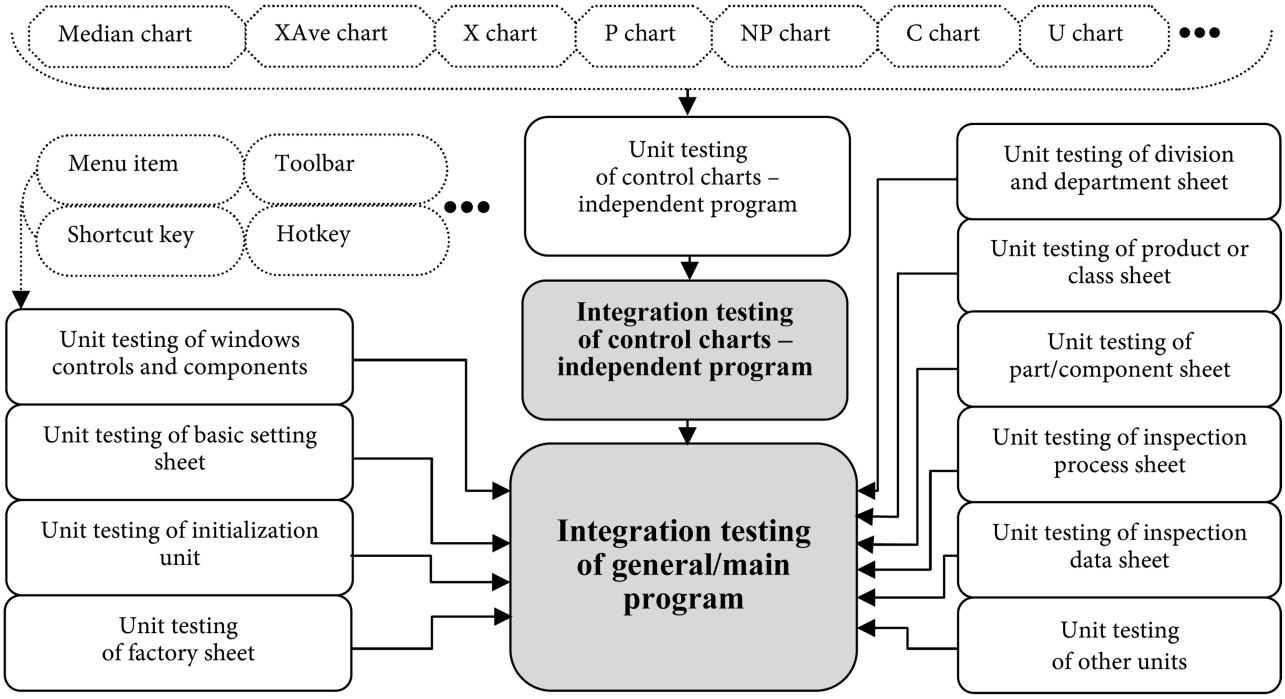

Figure 7. Scheme of integration testing with the improved Sandwich method.

high-level testing can be called as "GUI-oriented software testing", but it is noticed that this GUI testing should include the testing of member function driven by the GUI event. In a factual software system, there are various GUI controls and components, and a kind of control or component has also many units or instances, and this situation has lead to the difficulty of software testing including the organizing of testing activity. In order to solve this difficulty, the testing method based on "Sheet/Form" is put forward, and its main point is that the "Sheet/Form" is taken as a unified entity to execute unit testing coordinating with all controls and components in the "Sheet/Form". As a consequence, grey-box testing approach and improved STD method will be discussed in detail in the following.

1) "Grey-box approach".

a) Principle

In GUI software, we have known that various GUI controls and components concurrently drive the execution of a software function and many controls or components activate a software function by one event. For this situation, in the integration testing, if all integrated routes of all GUI controls and components are tested, the amount of testing work will be vast. That is to say, if all integrated routes from *Ci* to *Hi* should be tested, it will be a combination of "*Ci* (i = 1, 2, ···, *m*) × *Hj* (j = 1,2, ···, *n*)". For example, in a process *K* of one software function, if the number of event activating is U_K and the number of handling route is V_K, the test number of integrated routes for this software function *K* will be $U_K \times V_K$, and for $K = 1, 2, \cdots, F_P$, the test number of integrated route in integration testing will reach $\sum_{k=1}^{F_P} U_K \times V_K$ (Figure 8).

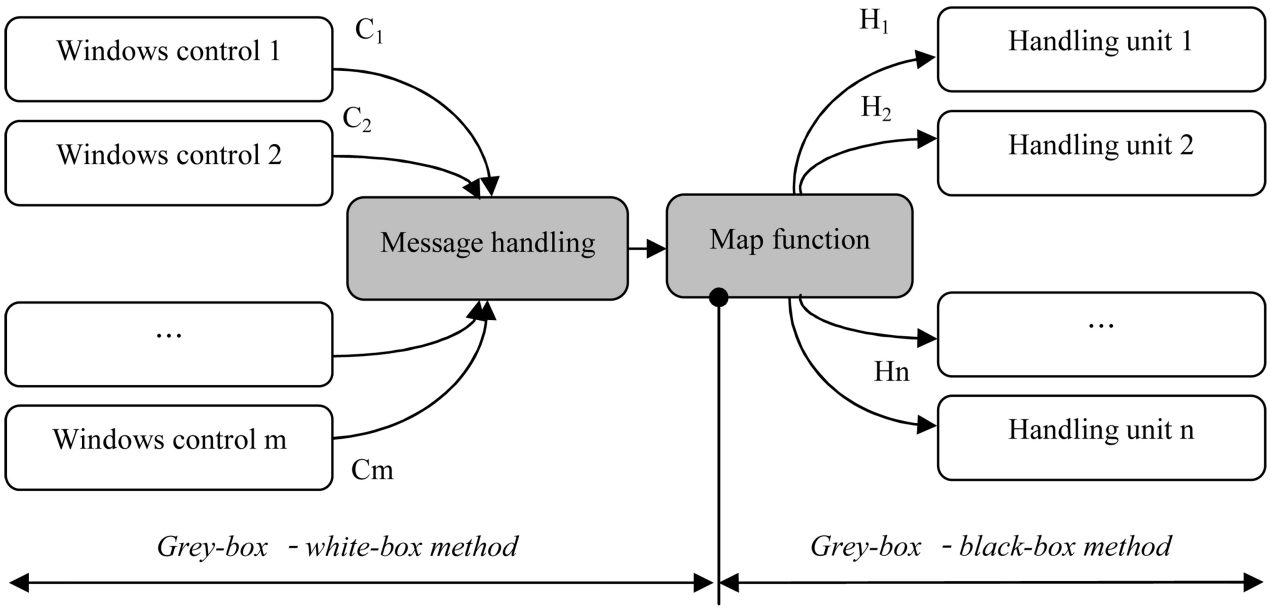

Figure 8. The principle of "Grey-box approach" in integration testing for GUI software.

Hence, in order to decrease the test number of integrated routes, we proposed a measure to deal with this problem, and it is the testing strategy of "Grey-box approach" [7]. In the "Grey-box approach", the pole for output of testing information is inserted into the map function, and for a process of software function with *m* activating event and *n* handling routes, we firstly test the correctness of the front part with white-box method, *i.e.* starting at "Windows control 1, …, Windows control *m*" and ceasing at mark "Point" of the pole, and then the follow-up part, *i.e.* from the pole to "Handling 1, …, Handling *n*", is tested with black-box method in sequence. And details of this approach may refer to [7] and [17].

Generally, the selection of pole position has several statuses for inserting into the mapping function, details are specified as follows.

- The path aggregation point across white-box analysis,
- The point that integrated route must go through,
- The entrance point of map function, and,
- The entrance point of initial member function.

b) Applying procedure

As mentioned above, the "Grey-box approach" is synthetic testing strategy that white-box method used in the fore-end to find out the error and BUG as early as possible with the message-handling mechanism, and the black-box method is applied to solve the testing process of more difficult follow-up part. As such, this approach is suitable to dispose the combination explosion problem for the majority of GUI software, including all kinds of application software developed with visual programming tools such as visual C++, visual Java, Delphi, etc. However, its effectiveness will be better for the software with more message types and more handling events.

Without loss generality, the procedure of test case constructing with the "Grey-box approach" may be demonstrated as follows.

- For one function disposing, several test cases of fore-end white-box testing should be constructed according to all kinds of control types.
- In terms of factual software, using black-box testing method, test cases of follow-up function testing are conducted distinguishing with various running situation.
- Conducting test case for all follow-up function disposing.
- In the process of follow-up construction of test case, the mapping function may be chosen according to the most rapid execution.
- If it is necessary, the test case of fore-end white-box testing must be not omitted for follow-up construction of test case.
- Necessary description should be given in the process of test case construction.

2) Improved STD (State Transform Diagram) method

a) Definition of improved STD

Facing to features of GUI software, the improved STD is proposed to deal with the function and state testing of GUI software. The difference of improved STD includes three aspects. (a) The symbol "●" in diagram presents the start point of software behavior, because a concrete state of control and event is existed to start for a state transforming in the GUI software, (b) the end point of diagram is labeled by symbol "⊙" for its graphical representation and expressiveness, and (c) the synthesis of many same or similar states is reasonably done. [2] [10]

b) Testing procedure based on improved STD

The procedure of function and state testing with improved STD could be given as follows.

- A programmer, acted as a tester in cross-testing, receives the finished code/program from another programmer, and does necessary analysis including FTA.
- According to the factual software, test design of software functions is firstly performed in detail, and taken care of the respective disposing based on actual running and exception handling.
- In terms of the front section above, the improved STD is drawn for the factual software.
- By the finished STD, test cases for state testing are consequently constructed.

Step 3 Testing executing.

The testing execution is a very important part in software testing activity, because it supplies the test result of a factual software system. Additionally, it is the basis of upgrading for software producing and testing. However, the testing execution in software testing organization should be done in terms of the factual situation, including the status of software system, the situation of the software company, and the status of employee, etc.

Here, we mainly discuss the testing execution based on "Grey-box approach" by "Cross-testing" organizing for "Pair-wise" mode, and the main process is shown in Figure 9. In this case, the integration testing should focus on function testing and state testing because the data testing generally has been finished in unit testing.

In the "Cross-testing", two programmers were be organized as a pair, and they manually cooperated for accomplishing the programming and testing, which one engaged in software programming and another one execute the software testing of the finished program by the former.

In testing execution, testing record must be done according to institution and arrangement of manager. Consequently, testing record should be updated with the alternation of test cases and test suite, and it is also the requirement to track BUG.

Step 4 Report and tracking.

The test report is the evidence of testing activity, and it is also the data source to track BUG and to look into the responsibility. Hence, the test report must be true, completed and reliable.

The trace of BUG is long-term and sustainable process, and it is necessary to use the statistic method and tool in the process of BUG tracking.

5. Result of Analysis and Tackling

5.1. FTA for Incremental Testing in Integration Testing

The incremental change happened typically in the upgrading of software version, but the self-constructing process of baseline version may be also considered as an incremental process. According to the view of software process management, however, the whole incremental testing based on the baseline version is more significant for update software products including all kinds of application

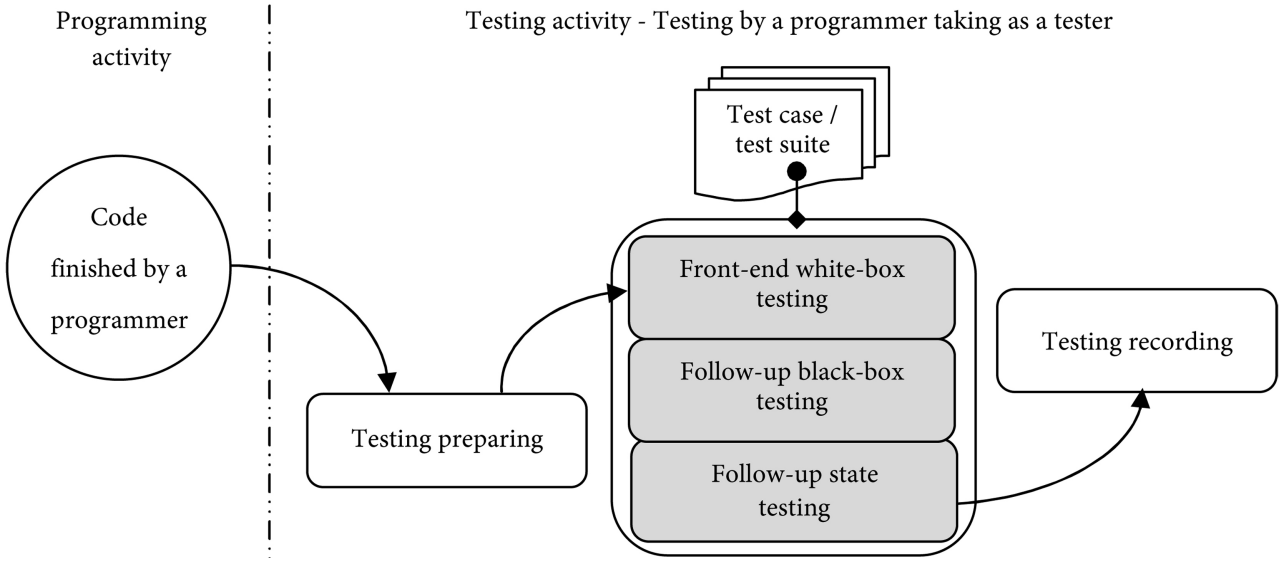

Figure 9. The testing execution for "Grey-box approach" by "Cross-testing" organizing.

areas, e.g. the student achievement management system, the factory quality control system, the hospital information management system, and tickets ordering system for air-line/online instamatic system, etc. Hence, following analysis will discussed the whole incremental testing based on the baseline version.

Primarily, FTA is an effective analysis tool of fault diagnosis for hardware system. Considering the clearness of technology roadmap and the easy-to-understand of technology principle, we applied this tool in the software testing especially for regression testing. By testing practice, we conclude that the FTA can be used for regression testing including the change of modifying and the change of adding. As such, a brief example will be investigated as illustrated in following, and more details can refer to [6].

As a consequence, for the testing organization of "Pair-wise" mode, if the incremental change happed, the incremental construction of regression test suite should be design by crossing tester, which this task is recommended to be done by skilled testing engineer, because the FTA must be done with skills. Without loss of typicality, we discuss the incremental change of adding.

According to the information derived from investigation and experience of quality engineers, the coordination of R chart should be independent in XAve-R chart of PQMS2. Thereupon, adding the coordination-offset coefficient for R chart is necessary respectively in XAve-R chart. Before the construction of regression test suite, the dependency analysis must be done to reveal the mutual relationship and influence conducted by programming change. Fault-tree analysis is very important tool for testing engineer, and **Figure 10** has illustrated the fault-tree of adding the coordination-offset.

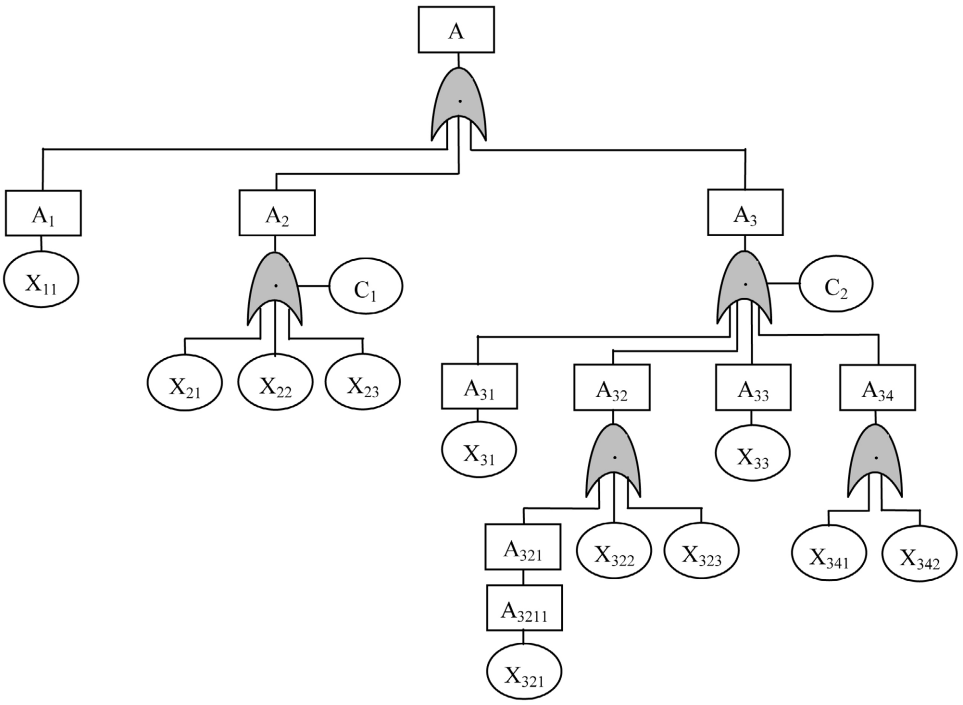

Figure 10. The fault tree of adding the coordination-offset for R chart.

In Figure 10, A is the top event, and A_i, A_{ij}, A_{ijk}, A_{ijkl} are the middle event respectively, and X_{ij}, X_{ijk} are the final event respectively, while C_i is the condition of event happening. As such, the detail description of the top and middle event is shown in Table 4, and the final event of fault-tree analysis is given in Table 5.

In terms of the results of fault-tree analysis derived from all final events, the requirement of constructing test case is list in Table 6, and detail process may refer to [6].

5.2. Test Suite Construction

In general, the software of integration testing can be typically divided into small scale program and large scale software by the view of scale. Moreover, on the

Table 4. The top and middle event of FTA.

Code	Event statement	Code	Event statement
A	Adding coordination-offset coefficient for R chart is necessary respectively in XAve-R chart	A_{33}	Disposing for GUI influence in OnDraw()
A_1	Adding control in GUI and variables in .cpp	A_{34}	Adding data interface parameters
A_2	Adding member in View class, Dialogue class and their objects	A_{321}	Disposing in global initialization
A_3	Adding member in data interface	A_{3211}	Disposing for influence—OnProcessUpdate()
A_{31}	Disposing for data saving interface	C_1	X_{21} is done before X_{22}
A_{32}	Disposing for data gathering interface	C_2	A_{34} is done before A_{33}, A_{32} and A_{31}

Table 5. The final event of FTA.

Code	Event statement	Code	Event statement
X_{11}	Adding Controls of Static Text, EditBox and their variables	X_{322}	Disposing of data getting in OnDraw()-GetSetting()
X_{21}	Declaration of "int RChart_offset"	X_{323}	Disposing of data getting in OnUpdate()-GetSetting()
X_{22}	Disposing in dialogue initializing-InitDialog()	X_{33}	Disposing for GUI influence in OnDraw()-RChart_offset
X_{23}	Disposing in View class-OnSetfigure()	X_{341}	Adding definition of unit in strSetting[]
X_{31}	Disposing in data saving-OnSaveSetting()	X_{342}	Adding disposing of chart initialization for definition of unit in strSetting[]
X_{321}	Disposing for influence in calling-OnGetCurrentProcess()		

Table 6. Choice[a] and adding[b] of test case in terms of result of FTA.

ID	Testing contents	Referring to
PQMS2-ENT-INT-TC304-MF	Add division and department from sheet	X_{341}
PQMS2-ENT-INT-TC307-MF	Add division and department to monitoring category	X_{341}
PQMS2-ENT-INT-TC314-MF	Add product from sheet	X_{341}
PQMS2-ENT-INT-TC324-MF	Add part from sheet	X_{341}
PQMS2-ENT-INT-TC327-MF	Add product_part to monitoring category	X_{341}
PQMS2-ENT-INT-TC334-MF	Add inspection process from sheet	X_{341}
PQMS2-ENT-INT-TC337-MF	Add inspection process to monitoring category	X_{341}
PQMS2-ENT-INT-TC360-MF	Input test data with saving manuallyfrom sheet	X_{341}
PQMS2-ENT-INT-TC906-MF	Display and preview XAve-R chart from monitoring category	X_{11}, X_{21}, X_{322}, X_{341}, X_{342}

a. Unit testing is not considered for choice here, b. Adding test case in unit testing—PQMS2-ACF-UNI-TC001~TC025-MF, etc.

view of software development, the software of integration testing can be differentiated into software developing based on baseline version and software updating for incremental change. Hence, the disposing of test suite constructing must consider all these distinctions.

5.2.1. Test Suite Constructing of Integration Testing—For Small Scale Program

For industrial practitioners, there is some simple application software, sometimes being called simple program, and it accomplished few functions even single function for particular requirement. In this case, the integration testing must execute in terms of factual situation, e.g. some testing items may be omitted for "unimportant integration" part as Table 2 under supervision of manager.

Somehow, the XAve-R chart program in PQMS2 may be taken as an example of small scale program, because its functions are relatively fewer to merely display and update the XAve-R chart.

1) Baseline version

Similarly, the integration testing of XAve-R chart program must be arranged after all related unit testing are finished, in which the "Triple-step method" should be adopted. For unit testing of XAve-R chart program in PQMS2, the data testing should mainly focus on the boundary value testing of input controls in the "Supervision or tolerance setting" sheet, the "Coefficient setting" sheet, and the "Layout parameter setting" sheet, and care should be taken for the data interface and format testing of text file of inspection data. At the same time, the

unit state testing should be done using the improved STD.

Because of having some window access control, the XAve-R chart program should execute the integration testing applying "Grey-box approach". As a consequence, test suite of front-end white-box for function testing should be constructed firstly. In order to shorten the demonstration, the function of "Layout parameter setting" is only given here, and examples are shown in Table 7.

Additionally, the integration function testing of XAve-R chart program should include integration function testing and integration state testing. The integration function testing mainly includes the function testing of initializing display, redisplay after coefficient and parameter are altered, and update displaying of XAve-R chart. The test case example of follow-up black-box of integration function testing for XAve-R chart is shown in Table 8.

As a small scale program, the integration state testing may be done similarly using the improved STD and it is not very complex. Furthermore, the improved STD of integration state testing of XAve-R chart is shown in Figure 11. In this diagram, S_0 is the initial state, and S_{12} is the end state, and "$e_{1/r1}$ $a_{1/r1}$, $e_{2/r2}$ $a_{2/r2}$, $e_{3/r3}$ $a_{3/r3}$, $e_{6/r6}$ $a_{6/r6}$, $e_{7/r7}$ $a_{7/r7}$, $e_{17/r17}$ $a_{17/r17}$, $e_{18/r18}$ $a_{18/r18}$, $e_{19/r19}$ $a_{19/r19}$, $e_{20/r20}$ $a_{20/r20}$, $e_{21/r21}$ $a_{21/r21}$, $e_{27/r27}$ $a_{28/r28}$, $e_{29/r29}$ $a_{29/r29}$" are bi-direction transformations, and it is noticed that S_4 is equivalent to S_0 mainly due to decrease the complexity of the diagram, and the specific meaning of these codes are omitted here.

According to the finished improved STD, the test suite of integration state testing for XAve-R chart is shown in table A_1 and A_2 of Appendix. For this suite, it is noticed that the testing steps have been depicted in detail to operate conveniently for testing execution.

2) Incremental testing

In previous "Layout parameter setting" sheet of XAve-R chart program, "offset of coordinate axle" has been given, but "offset of R coordinate axle" is not given, so that the adjusting of R chart has a bit of difficulty. In PQMS2, it is quite necessary to add the parameter of R coordinator-offset in the "Drawing layout parameter setting" sheet as shown in Figure 12.

Table 7. Example of test cases of front-end white-box for the XAve-R chart program.

Precondition—Insert the pole of < MessageBox ("Testing output."); > in the front of member function <void CEXE9_7View::OnSetfigure()>.		
ID	Input	Expected output
PQMS2-AFW-INT-TC100-AD	In the XAve-R window interface, click the menu item of "Setting—Drawing layout", and activate the "Layout parameter setting" sheet.	Prompt "Testing output."
PQMS2-AFW-INT-TC101-AD	Click the shortcut key "Alt-B" and "Alt-P" from the menu item of "Setting (**S**)—Drawing layout (**F**)".	Prompt "Testing output."
PQMS2-AFW-INT-TC102-AD	Click the toolbar item of "Layout", and activate the "Layout parameter setting" sheet.	Prompt "Testing output."

Table 8. Example of test cases of black-box for the XAve-R chart program.

Precondition—Delete the pole of <MessageBox ("Testing output.");> in the front of member function <void CEXE9_7View:: OnSetTolerance(), void CEXE9_7View:: OnSetScale(), void CEXE9_7View:: OnSetfigure(), void CEXE9_7View:: OnXAveChartMonitorUpdate(), void CEXE9_7View:: OnXAveChart MonitorRevUpdate(), void CEXE9_7View:: OnUpdate()>.		
ID	Input	Expected output
PQMS2-ACF-UNI-TC100	Start the XAve-R chart with default setting and inspection data, and values of default setting are given with "strSetting [20] = {220, 750, 0.60, 0.46, 10.0, 220, 1000, −530, 1000, 11, 30, 820, 1.25, 0, 1500, 130, 10.0, 0.75, −340, 4}", and inspection data is the data batch of "2018_01_01 INSPDAVA2_1 4" with 80 values.	Display precisely with correct layout and no prompt of error information.
PQMS2-ACF-UNI-TC010	Start the "supervision/tolerance setting" sheet from the XAve-R window interface of the XAve-R chart, and input "0.60" in the EidtBox "Supervising or tolerance lower limit", and input "0.46" in the EidtBox "Supervising or tolerance up limit", and input "4" in the EidtBox "Sampling volume", and click button "OK" finally.	Update displaying precisely with correct layout and no prompt of error information.
PQMS2-ACF-UNI-TC001	Start the "Coefficient setting" sheet from the XAve-R window interface of the XAve-R chart interface, and input "10.0" in the EidtBox "Vertical coordinate magnitude coefficient", and input "1.25" in the EidtBox "Total magnitude coefficient", and input "10.0" in the EidtBox "R vertical coordinate magnitude coefficient", and input "0.75" in the EidtBox "R total magnitude coefficient", and input "0" in the EidtBox "Update velocity", and input "1500" in the EidtBox "history displaying space", and click button "OK" finally.	Update displaying precisely with correct layout and no prompt of error information.
PQMS2-ACF-UNI-TC020	Start the "Layout parameter setting" sheet from the XAve-R window interface, and input "11" in the EidtBox "Caption offset", and input "1000" in the EidtBox "Right space", and input "820" in the EidtBox "Limits offset", and input "1000" in the EidtBox "y offset", and input "220" in the EidtBox "x offset", and input "30" in the EidtBox "Judgment output", and input "−530" in the EidtBox "Coordinate axle offset", and input "750" in the EidtBox "Ry offset", and input "220" in the EidtBox "Rx offset", and input "130" in the EidtBox "R judgment output", and click button "OK" finally.	Update displaying precisely with correct layout and no prompt of error information.
PQMS2-ACF-UNI-TC101	After TC100, click menu item "Monitoring of XAve-R—Forward" or toolbar "Monitoring forward".	Display next chart precisely with correct layout and no error prompt.
PQMS2-ACF-UNI-TC102	After TC101, click menu item "Monitoring of XAve-R—Backward" or toolbar "Monitoring backward".	Display previous chart precisely with correct layout and no error prompt.
PQMS2-ACF-UNI-TC103	Click menu item "Monitoring of XAve-R—Update" or toolbar "Monitoring Update".	Update displaying of current chart precisely with correct layout and no error prompt.

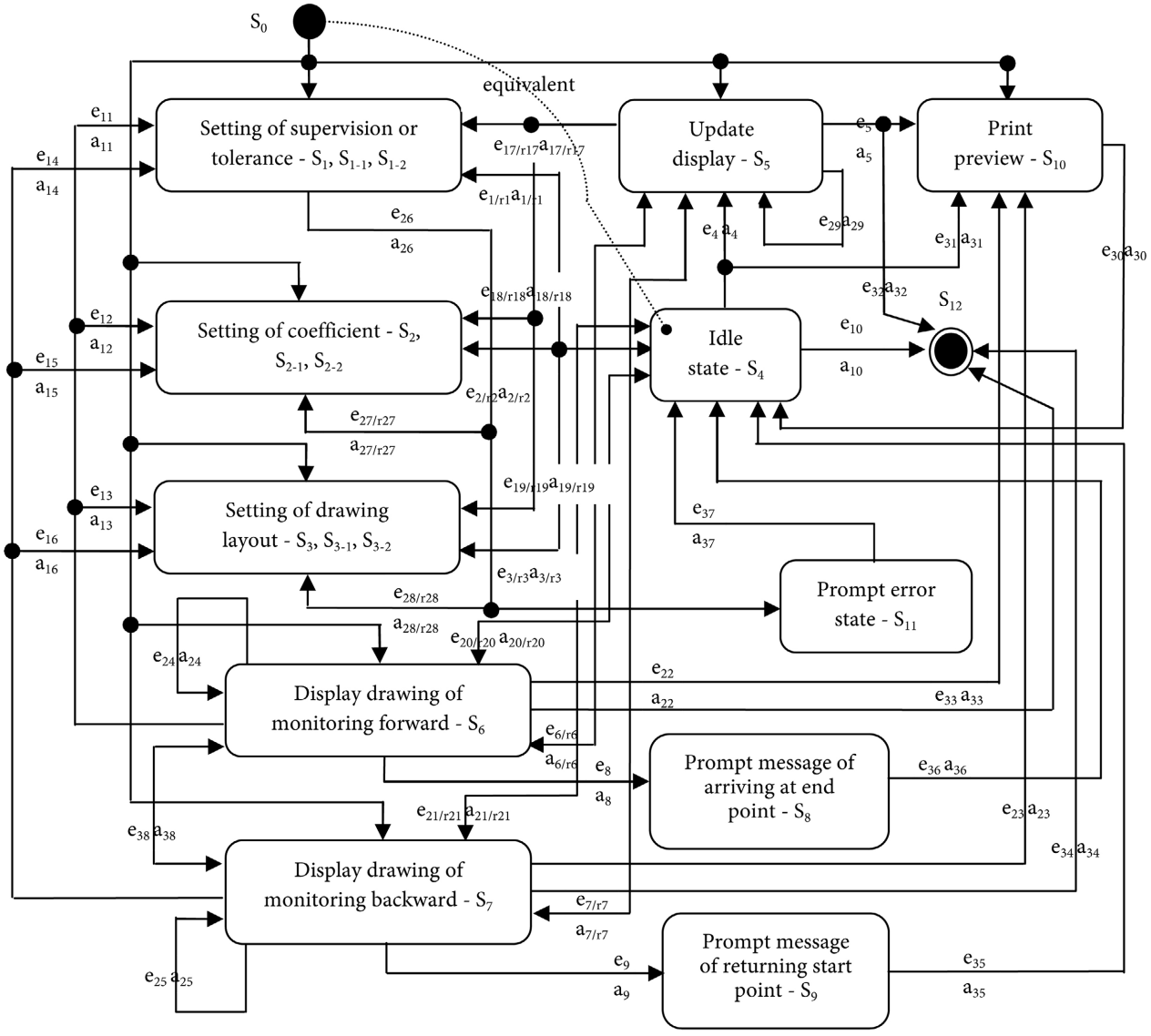

Figure 11. The improved STD of integration testing for XAve-R chart program.

For this incremental change of adding "offset of R coordinate axle", the incremental unit testing of this "Layout parameter setting" sheet must be executed firstly including (a) data testing—mainly data boundary value testing, (b) function testing—such as function-self running testing and other limitation testing, and (c) state testing—test suite should be constructed in terms of the improved STD which is relative very simply.

For this incremental integration testing, as a kind of small scale program, data testing is unnecessary to be repeated, and function testing must be done with new test case which derived from the result of FTA and can be modified using "PQMS2-ACF-UNI-TC020" listed in **Table 8** above, and the detail of test case of the function incremental testing is shown in **Table 9**.

Additionally, because this small scale program—XAve-R chart program was integrated into the whole system—PQMS2, the follow-up integration testing for

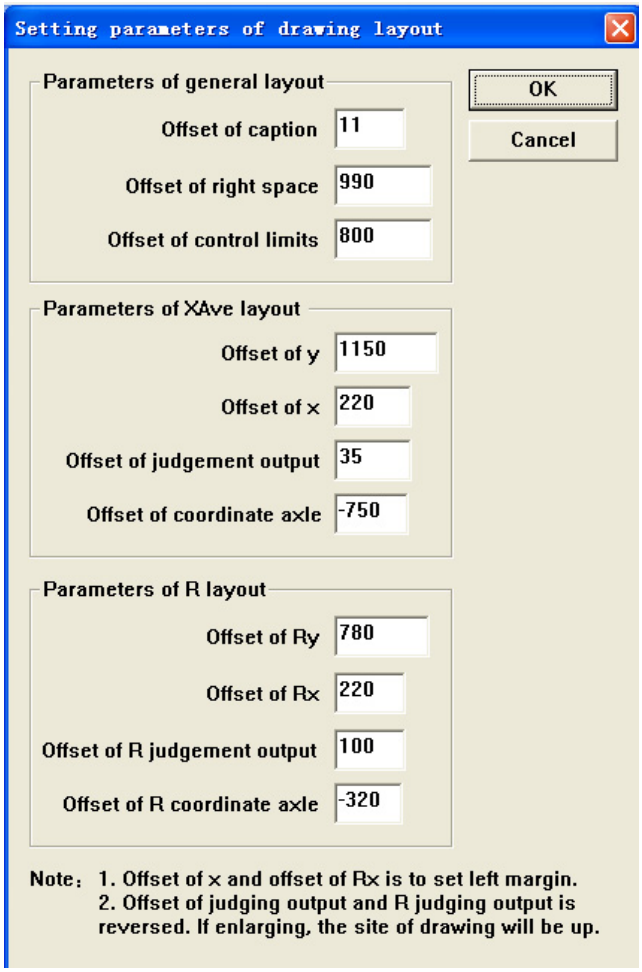

Figure 12. GUI of drawing layout setting sheet.

PQMS2 must be done in terms of "Modified Sandwich" method [4]. At the same time, we can directly use the result of FTA as mentioned in Table 6 above, and the test case with the code of "PQMS2-ENT-INT-TC906-MF" is given in Table 10.

5.2.2. Test Suite Constructing of Integration Testing—For Large Scale Software

1) Baseline version

The test suite construction of integration testing of baseline version, such as "Main program", should be done generally using "Grey-box approach", except that it is simple program or non-GUI software as mentioned above. For the constructing of test suite of integration testing based on "Grey-box approach", the general procedure can be depicted as follows.

a) For every function disposing, several test cases of fore-end white-box testing should be constructed according to all kinds of window control types.

b) In terms of factual software system, using black-box testing method, construct test cases of function testing distinguishing with various running situation.

Table 9. Incremental test case of the "Drawing layout parameter setting" sheet.

Precondition—Delete the pole of <MessageBox ("Testing output.");> in the front of member function of <void void CEXE9_7View:: OnSetfigure()>.		
ID	Input	Expected output
PQMS2-ACF-UNI-TC020-MF	Start the "Drawing layout parameter setting" sheet from the XAve-R chart interface, and input "11" in the EidtBox "Caption offset", and input "1000" in the EidtBox "Right space", and input "820" in the EidtBox "Limits offset", and input "1000" in the EidtBox "y offset", and input "220" in the EidtBox "x offset", and input "30" in the EidtBox "Judgment output", and input "−530" in the EidtBox "Coordinate axle offset", and input "750" in the EidtBox "Ry offset", and input "220" in the EidtBox "Rx offset", and input "130" in the EidtBox "R judgment output", and input "−340" in the EidtBox "R coordinate axle offset", and click button "OK" finally.	Update displaying precisely with correct layout of "offset of R coordinate axle" and no prompt of error information.

Table 10. Incremental test case of integration function for the whole PQMS2.

Precondition—Delete the pole of <MessageBox ("Testing output.");> of QMS2-ENT-INT-TC350-AD~TC353-AD, and finish data input of the inspection data item "Division of machining and cutting—CM_Digital thickness inspection_Lock pollar-thickness-2019_09_18 INSPDAVA2_1 4" in the monitoring category.		
ID	Input	Expected output
PQMS2-ENT-INT-TC906-MF	Start the main window interface of PQMS2, and click the inspection data item "Division of machining and cutting—CM_Digital thickness inspection_Lock pollar-thickness-2019_09_18 INSPDAVA2_1 4" in the monitoring category, and click button "Do monitoring-2_XAvechart".	Displaying precisely XAve chart with correct layout and no prompt of error information.

c) Conducting test case for all follow-up function disposing if necessary.

d) In the process of follow-up construction of test case, the mapping function may be chosen according to the most rapid accuracy.

e) Necessary description should be given in the process of test case construction.

Here, the constructing method and writing format are given for the integration testing using the example of PQMS2. Without loss generality for GUI software, the integration testing of the "Part/component" sheet is discussed according to the "Grey-box approach", and the GUI of "Part/component" sheet in PQMS2 is shown in Figure 13.

The "Part/component" sheet is typical GUI in PQMS2, which includes popular GUI controls and components, e.g. "Button" control, "Editbox" control, "ComBoBox" control, and "List" component, etc. At the same time, this "Part/component" sheet is activated by message mechanism of the menu item in main system interface and all functions in this sheet are also activated by the

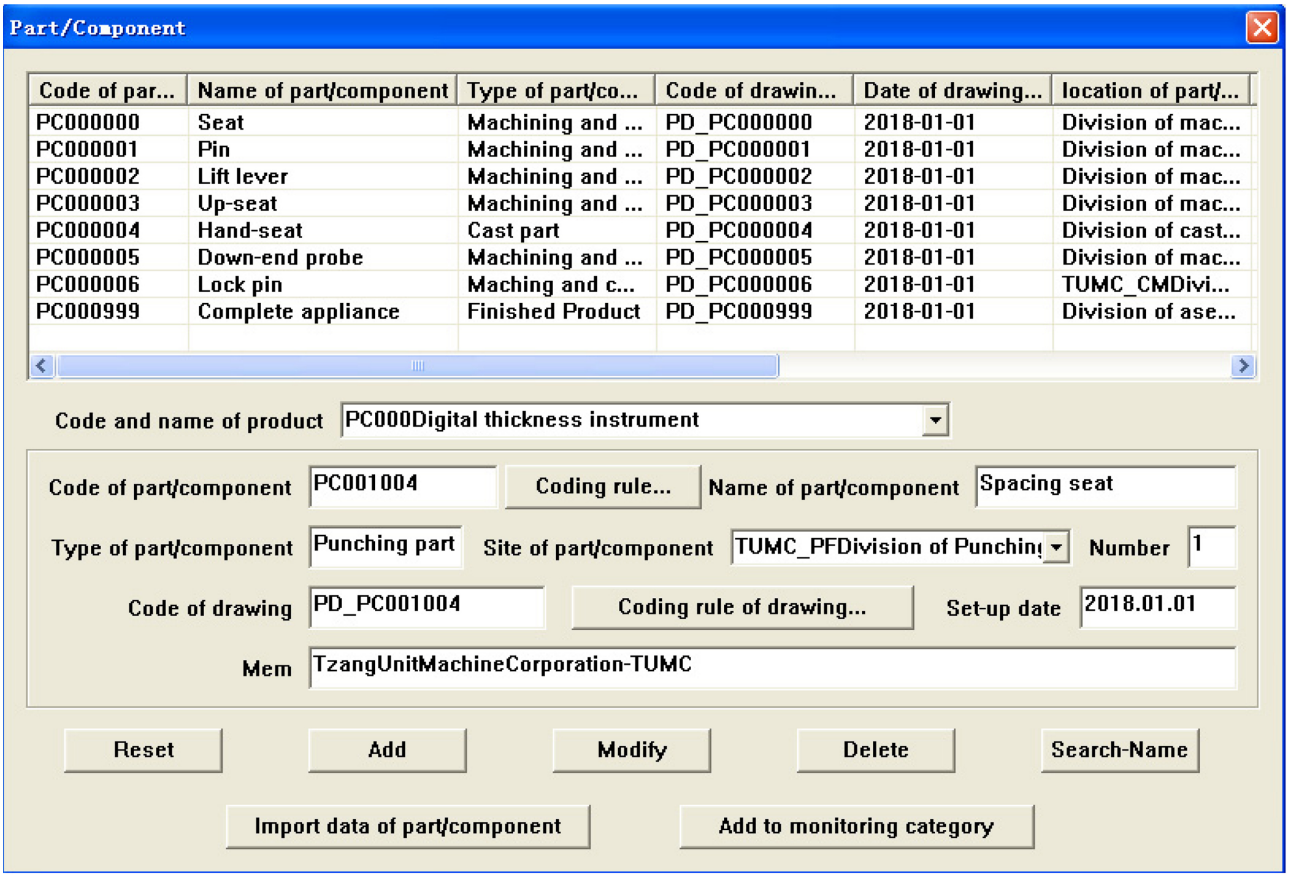

Figure 13. The GUI of "Part/component" sheet in PQMS2.

event of controls and components. Hence, the test suite of this sheet is constructed based on the "Grey-box approach".

According to the applying procedure of "Grey-box approach", the front-end test suite is constructed with white-box method and the follow-up test suite is conducted by black-box method in the "Grey-box approach". Consequently, the test suite of front-end white-box is given in Table 11, and the test suite of follow-up black-box is given in Table 12.

In writing of test case for "Grey-box approach", following matters should be noticed.

- For construction of the test suite of front-end with white-box, the precondition should be given in terms of actual site for inserting the testing probe, and all kinds of control types to activate the message event must be given without omission.
- For construction of the test suite of follow-up with black-box, the test case of unit testing should be not repeated, and the test case of integration testing between units should be constructed.

2) Incremental testing

Besides the test suite construction of integration testing for baseline version, the test suite construction of incremental change will occur in integration testing too, and the incremental testing generally is probably due to the change with

Table 11. Test cases of adding part/component in "Part/component" sheet—white-box.

Precondition—Insert the pole of <MessageBox ("Testing output.");> in the front of member function of <BOOL CProductPartDIALOG::OnInitDialog()>.		
ID	Input	Expected output
PQMS2-ENT-INT-TC320-AD	In the left monotoring category of main window interface, click the item "Division of machining and cutting", choose "Add product_part/component" using the right key of mouse, and activate the "Part/component" sheet.	Prompt "Testing output."
PQMS2-ENT-INT-TC321-AD	In the main window interface, click the menu item of "Basic data of product quality—Part/component", and activate the "Part/component" sheet.	Prompt "Testing output."
PQMS2-ENT-INT-TC322-AD	Click the shortcut key "Alt-B" and "Alt-P" from the menu item of "Basic data of product quality (**B**)—Part/component (**P**)".	Prompt "Testing output."
PQMS2-ENT-INT-TC323-AD	In the main window interface, click the toolbar item of "Component" , and activate the "Part/component" sheet.	Prompt "Testing output."

Table 12. Test cases of adding part/component in "Part/component" sheet—black box.

Precondition—a. Delete the pole of <MessageBox ("Testing output.");> in the front of member function of <BOOL CProductPartDIALOG::OnInitDialog()>. b. the item of "PC000006-Lock pin—Machining and cutting part—…" has existed in the list of "Part/component" sheet but not included in the monitoring category.		
ID	Input	Expected output
PQMS2-ENT-INT-TC327-MF	In the "Part/component" sheet, choose the item "PC000006-Lock pin—Machining and cutting part—…" of the list, and click the button "Add to monitoring category" finally.	Display the information prompt of finished adding, and update the data in the monitoring category.

cross-influence for several units. If the incremental integration testing is dealt with this kind of change of cross-influence, the dependency analysis is necessary and the FTA can be usually applied as mentioned above; consequently, the construction of the test suite of front-end with white-box and follow-up with black-box should done using "Grey-box approach" as mentioned above; and details are omitted here. Hence, we only give the example of modification of window access controls.

As we all known, window access controls and its implementation functions usually are needed to modify sometimes, e.g. for fulfilling the supervision of key sampling test or GUI check. If the window access control is modified, the test case of integration testing must be constructed again in terms of "Grey-box approach". As a typical example, in PQMS2, the menu item of "Output of inspection data" is modified to "Backup of inspection data output", and test cases of fore-end with white-box were changed with ID "PQMS2-ENT-INT-TC368-MF~TC369-MF" as shown in Table 13, but test cases of the black-box testing are unnecessary to modify again if without change

Table 13. Test cases for the menu item of "Backup of inspection data output".

Precondition—Insert the pole of <MessageBox ("Testing output.");> in the front of member function of <BOOL CExportDataDialog::OnInitDialog()>.		
ID	Input	Expected output
PQMS2-ENT-INT-TC368-MF	In the main window interface, click the menu item of "Data I/O—Backup of inspection data output", and activate the backup of inspection data.	Prompt "Testing output."
PQMS2-ENT-INT-TC369-MF	Click the shortcut key "Alt-T" and "Alt-I" from the menu item of "Data I/O (**T**)—Backup of inspection data output (**I**)".	Prompt "Testing output."

in implementation function.

5.3. Testing Executing

In the integration testing of GUI software, in terms of the test case and test suite, the tester executes the testing process as follows.

a) Checking whether the unit testing of all relative units is finished according to the requirement of standard and manager.

b) If the software is a system based on window controls, the grey-box testing approach should be adopted.

c) Execute the front-end testing of white-box test suite.

d) Execute the follow-up testing of black-box test suite.

e) If the test suite of state testing is existed, the state test should be done after the necessary familiar of the improved STD.

f) Necessary description should be given in the process of test case construction.

g) Recording test process necessarily and the test result in detail including the BUG with detail information.

Additionally, following maters should be paid attention.

a) At first, the correct and effective test case and test suite are needed before testing execution.

b) The desk check of code review must be done by programmer before integration testing.

c) The programmer and the tester can execute "Cross-testing" in terms of "Parallel-disposing" pattern and "Idle-waiting" pattern [2].

d) If the incremental testing occurred based on FTA, the testing should be executed by a skilled software testing engineer.

e) If the software is a network system with database, the integration testing of data interface and communication is required.

f) If it is necessary, the process boundary testing should be assigned.

g) If the requirement of software is rigid, the walkthrough and other audit should be finished as early as possible. At the same time, the function and state testing must be executed with "N-switch" consideration.

h) If it is necessary, the tester could feed back the reasonable modifying

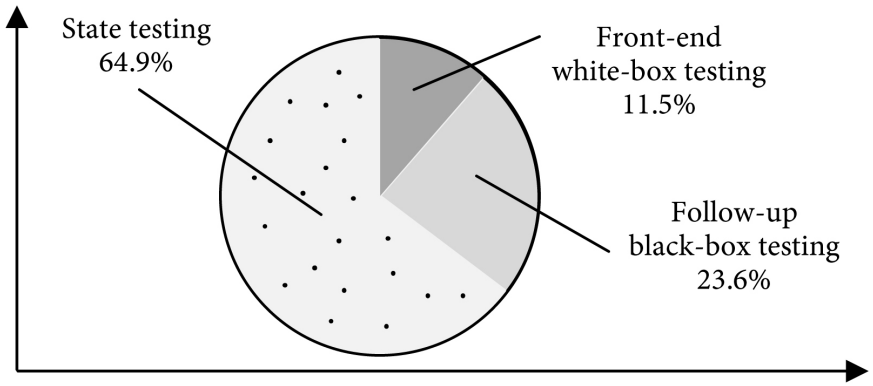

Figure 14. The pie chart of testing time in integration testing.

suggestion.

For testing execution of integration testing, according to the requirement of GUI software, we have performed the testing practice for PQMS2 with some experimental ways. Without loss representative and generality, the factual result of testing time of testing execution for the independent program XAve-R chart in PQMS2 are: (a) the testing time of front-end white-box testing is 2.63 min, (b) the testing time of follow-up black-box testing is 5.43 min, and (c) the testing time of state testing is 14.92 min. Consequently, the pie chart of this example is illustrated in **Figure 14**.

From **Figure 14**, we can find that the testing time of state testing had occupied 64.9%. Hence, we suggest that the advanced testing technology and method should be invented and adopted to accelerate the process of state testing.

5.4. Report and Tracking

In generally, the test report must be correct and precise without mistake for misunderstanding. In testing activity for GUI software, the test report format could directly be generated from the test case with table style, while the BUG state must be recorded as factual situation. The following must be noticed for the test report.

- Data of test report must be true, completed and reliable.
- Results of test report must be written in terms of the evidence of testing activity.
- Conclusion must be given without being ready to accept either course.
- Test report must include the column of the BUG state written clearly by tester as concise as possible.

In order to implement BUG tracking, the BUG recording must be saved completely as possible. If necessary, the BUG database should be developed to achieve a long-term and sustainable control.

6. Discussion

The main implication of our presented work in this paper is that good strategy

and methodology will greatly improve the efficiency and quality of integration testing for GUI software, and avoid the redundant operation in integrating testing activity, including the adoption of effective testing organization. However, care must be taken what kind software system the tested object is and how the testing activity is organized for actual software companies including personal composition, e.g. product-oriented organizing or project-oriented organizing.

6.1. Special Disposing Methods for Particular Software System

6.1.1. Special Disposing for Embedded Software

In major industrial practice, a lot of software exists in the form of embedded software. Usually, majority of software in industry is a part composed of instrument and equipment, in this case, the factual software may be considered as the embedded software. The testing of this kind of embedded software must be disposed in terms of its own features as follows.

- Stub technology usually used in embedded software testing.
- The performance testing must be emphasized, including memory utilization testing, I/O testing, etc.
- Logic testing should be prior to apply complier collection mode, and using perfect test tools is a good choice.

6.1.2. Special Disposing for Inspection Software

For inspection software, including instrument software, the initialization is a very important part for the whole software system, because good initialization will improve the starting of software system. At the same time, the setup part of software system should be rigidly considered, especially concerning the factual applied situation of software system. Historically, the serious failure of software setup in the instrument had got rise to terrible result. In 1985-1987, the Therac-25 instrument made by Atomic Energy of Canada Limited brought about 5 dead cases, and all accidents are caused by the wrong operation and software fault of error-setup disposing [4]. Hence, the testing of basic setting and initialization units must be paid more attention.

6.2. Threats of Validity

In order to assure the validity of study results including the feasibility of strategy and methodology of integration testing for GUI software, several measures are taken. In general, the part of strategies is concluded from mass experience of software testing practice with a view of scientific conclusion, and the part of methodology is illustrated and proved with a lot of factual examples.

6.2.1. Internal Validity

At first, to assure the representative, factual examples were carefully chosen from the GUI software—PQMS2 to avoid the bias of lack of functionality and GUI interface, which includes most of general GUI controls and components and runs under the typical Windows system. At the same time, the integration test-

ing activity of PQMS2 has the representative of most GUI application software with features of more functionalities and more interaction of "Forms/Sheets". Further, the testing execution of PQMS2 has also included the typical achievement of the integrated process of the modified Sandwich method.

On the other hand, in this study, for the methodology of integration testing for GUI software, we investigate two main research methods and tools, *i.e.* the FTA (Fault Tree Analysis) method and the improved STD (State Transform Diagram) method. FTA is mainly used in coding and programming analysis for incremental testing, and the improved STD is mainly applied for function and state testing.

FTA is wonderful analysis tool for software failure and BUG derived from fault diagnosis. For BUG produced after software modification change, it is effective tool in terms of the factual result from fault analysis in engineering application. For unit adding or BUG hypothesis generated from software addition change (BUG hypothesis is better disposing for the choice of logic gate), it is also effective method because this analysis diagram can clearly and precisely describe the unit composition of software and the relationship among software units, and fault and BUG derivation can be conveniently done.

However, when FTA is applied to analyze the change of software addition, the effectiveness will rely on the precise definition of logic relationship among added units, and the accurate computation of cut-set etc. may be difficult. Hence, it will have some problems for this applied case and scenario in safety-critical system, but it is valuable and useful for the factual application in general software system.

The improved STD has its factual effectiveness for the improvement of diagram related to the software engineering feature including the expression of start point. If the division and layout of the improved STD is clear and correct as possible, the improved STD can effectively analyze the whole sight of all states and can effectively depict the specific feature of the divided layer and transforming route of the software running behavior. Additionally, in order to keep the clearness of description, we proposed a useful strategy which "the curved diagram" is used for the small scale case and "the straight-line diagram" is applied for the large scale case.

6.2.2. External Validity

At present, the GUI software for smart-phone is bloomed with the communication technology. The improved STD, used in desktop system and derived from desktop application, is a universal tool for software state analysis, and it can be also applied in the smart-phone software with a limited changing.

6.2.3. Construct Validity

In this study, on the one hand, specific strategy and method are derived from the summarization of software testing practice, and have been verified in factual testing process. As such, strategy and method proposed in this study will ensure

the validity to help practitioners of software testing. On the other hand, the hypothesis of "0-switch~N-switch state testing" with the improved STD for general GUI software has been verified for its validity using examples of PQMS, except the safety-critical software with "N-switch state testing".

6.2.4. Conclusion Validity

By the case software—PQMS as the representative and typical GUI software, research was performed using many examples including a lot of detail disposing. These specific disposing should strengthen the validity of this work with the doubtless replication of the study and factual application in similar contexts, e.g. the example of section 5.2.2.1 can be applied in "Sheet/Form" situation and the example of section 5.2.2.2 is fitted to the situation driven directly by window access control. Additionally, meaningful references have been added to clarify the strategy and methodology in this study, e.g. the "Grey-box" approach for integration testing of GUI software [7], the Improved STD for the function and state testing [2], etc.

7. Summary

This paper presents strategy and methodology of integration testing for GUI software, and discusses the integration testing activity for two situations including small-scale program and large-scale software. In detail, we proposed various strategies of integration testing for GUI software, including differentiating strategy of distinguished software system, testing organizing strategy, testing procedure strategy, and test suite construction strategy. These strategies could give the instruction for software testing practitioners to improve the testing efficiency and strengthen the process control of software testing activity.

Furthermore, we have deeply discussed a set of effective methods mainly focusing on test case and suite construction in integration testing, e.g. the testing arrangement method based on "Grey-box approach" for the modified Sandwich integration strategy. Facing the update evolution trend of software system, we investigate the test suite construction method based on baseline version and test suite construction method based on incremental change. Then, we also briefly describe the testing execution for the testing organizing of "Cross-testing" and the emerged problem within it. All methods would instruct practitioners to rapidly set up the testing process in terms of "Triple-step method" and "Grey-box approach" for GUI software.

This study thereby provides a contribution to the software companies about the effective strategies in integration testing for GUI software. At the same time, the study provides a contribution to the software testing practitioners about the specific execution method for integration testing based on "Grey-box approach".

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Alegroth, 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. https://doi.org/10.1007/s10664-016-9497-6

[2] TanLi, M., Zhang, Y. and Wang, Y.L. (2022) Architecture and Methodology of Unit Testing Embedding Pair-Wise Mode for Small Team. *Journal of Software Engineering and Applications*, **15**, 111-133. https://doi.org/10.4236/jsea.2022.1511022

[3] Patton, R. (2006) Software Testing. Pearson Education Inc., NewYork, USA.

[4] Fu, B. (2014) Course of Software Testing Technology. Tsinghua University Press, Beijing.

[5] Li, F. (2016) Software Testing Technology. Mechanical Industry Press, Beijing.

[6] TanLi, M., Zhang, Y. and Wang, Y.-L. (2020) Research on Fault Tree Technique in Software Regression Testing. *Computer Engineering and Software*, **41**, 5-8, 25.

[7] TanLi, M., Zhang, Y., Wang, Y.L., *et al.* (2021) Grey-Box Technique of Software Integration Testing Based on Message. *Proceedings of* 3*rd International Conference on Artificial Intelligence and Computer Science*, Beijing, 29-31 July 2021, 198-206.

[8] Runeson, P. (2009) Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, **14**, 131-164. https://doi.org/10.1007/s10664-008-9102-8

[9] Bradbury, J.S., Cordy, J.R. and Dingel, J. (2005) An Empirical Framework for Comparing Effectiveness of Testing and Property-Based Formal Analysis. *Proceedings of the ACM SIGPLAN-SIGSOFT Work on Program Analysis for Software Tools and Engineering*, Lisbon, 5-6 September 2005, 1-4. https://doi.org/10.1145/1108792.1108795

[10] Tang, D., TanLi, M. and Li, T. (2022) Software Test Organizing for Small Team Based on "Pair-Wise" Mode. *Proceedings of* 2022 *International Conference on Smart Transportation and Future Mobility*, Changsha, 2-4 September 2022, 113-119.

[11] TanLi, M., Zhang, Y., Jiang, Y., *et al.* (2021) Baseline Test Suite Construction of Smoke Test for Extreme Programming. *Proceedings of* 2021 *International Conference on Communication Engineering and Logistics Management*, Shanghai, 24-26 July 2021, 1-7. https://doi.org/10.1088/1757-899X/1179/1/012001

[12] TanLi, M., Zhang, Y. and Wang, Y.-L. (2020) System Testing Based on Software Performance. *Computer Engineering and Software*, **41**, 1-4, 25.

[13] Xu, Y.-Y. (2015) A Study of Test Case Reuse Based on CBR. *Computer Engineering and Software*, **36**, 117-120.

[14] TanLi, M., Jiang, Y., Wang, Y.L., *et al.* (2020) Infrastructure Building of Software Testing for Engineering Software Based on Cooperation of University and Company. *Proceedings of the* 10*th International Workshop on Computer Science and Engineering* , Shanghai, 19-21 June 2020, 18-26.

[15] Chen, Z.H. (2005) Research and Implementation of Test Method in Task Arrangement of Resource Satellite. *Radio Engineering*, **35**, 62-64.

[16] TanLi, M., Jiang, Y., Wang, Y.L., *et al.* (2018) Digital Inspection of Cutting and Machining Based on Manufacturing Quality for Shop Floor. *ICMEIT*2018, Shanghai, 23-24 April 2018, 1-7. https://doi.org/10.12783/dtetr/icmeit2018/23372

[17] TanLi, M., Xiao, J.Y. and Zhang, Y. (2023) Guideline of Test Suite Construction for GUI Software Centered on Grey-box Approach. *Journal of Software Engineering*

and Applications, **4**, 385-405. https://doi.org/10.4236/jsea.2023.165007

[18] Do, H., Rothermel, G. and Elbaum, S. (2004) Infrastructure Support for Controlled Experimentation with Software Testing and Regression Testing Techniques. *Proceedings of the* 2004 *International Symposium on Empirical Software Engineering*, Redondo Beach, 13 April 2004, 60-70.

[19] TanLi, M., Zhang, Y. and Wang, Y.L. (2021) Boundary Clarify between Integration Testing and Validation Testing and Engineering Example. *Proceedings of the* 3*rd International Conference on Computer Science*, *Communication and Network Security*, Sanya, 19 May 2021, 1-7. https://doi.org/10.1051/itmconf/20224501004

Appendix

Table A1. State test suite of valid one step or multiple steps transformation for XAve-R chart.

ID	Start state	Next/End state	Input	Expected output
PQMS2-ACS-INT-TC001	S_0	S_1-S_4-S_{12}	Start the XAve-R program interface, and click the menu item of "Setting—Supervision/Tolerance setting" or toolbar item of "Supervision/Tolerance", and enter "Supervision/Tolerance" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet, and return the XAve-R interface. Click the button "Shut off" in the up-right corner of the XAve-R interface.	Display "Supervision/Tolerance setting" sheet, and shut off. After return to the idle state, from the XAve-R interface, finally exit program.
PQMS2-ACS-INT-TC002	S_0	S_2-S_4-S_{12}	Enter the XAve-R interface again, and click the menu item of "Setting—Coefficient setting" or toolbar item of "Coefficient", and enter "Coefficient setting" sheet. Without input, click the button "Cancel" of the sheet, and return the XAve-R interface. Click the button "Shut off" in the up-right corner of the XAve-R interface.	Display "Coefficient setting" sheet, and shut off. After return to the idle state, from the XAve-R interface, finally exit program.
PQMS2-ACS-INT-TC003	S_0	S_3-S_4	Enter the XAve-R interface again, and click the menu item of "Setting—Drawing layout setting" or toolbar item of "Drawing layout", and enter "Drawing layout setting" sheet. Without input, c lick the button "Cancel" of the sheet., and return the XAve-R interface.	Display "Drawing la-yout setting" sheet, and shut off, then return to the idle state.
PQMS2-ACS-INT-TC004	S_4	S_1-S_4	After the XAve-R program idle interface, click the menu item of "Setting—Supervision /Tolerance setting", and enter "Supervision /Tolerance" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Display "Supervision /Tolerance setting" sheet, and shut off, then return to the idle state.
PQMS2-ACS-INT-TC005	S_4	S_2-S_4	Enter the XAve-R program idle interface, and click the menu item of "Setting—Coefficient setting", and enter "Coefficient setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Display "Coefficient setting" sheet, and shut off, then return to the idle state.

Continued

PQMS2-ACS-INT-TC006	S4	S3-S4-S12	Enter the XAve-R program idle interface, and click the menu item of "Setting—Drawing layout setting", and enter "Drawing layout setting" sheet. Without input, click the button "Shut off" in the up-right corner of "Drawing layout setting" sheet. Click the button "Shut off" in the up-right corner of the XAve-R interface.	Display "Drawing layout setting" sheet, and shut off, then return to the idle state. Finally, Exit XAve-R program.
PQMS2-ACS-INT-TC007	S0	S6	Click the menu item "XAve-R chart monitoring—Monitoring forward" in the XAve-R program interface.	Execute monitoring forward.
PQMS2-ACS-INT-TC008	S6	S7	Click the menu item "XAve-R chart monitoring—Monitoring backward" in the XAve-R program interface.	Execute monitoring backward.
PQMS2-ACS-INT-TC009	S7	S7	Continuously click the toolbar item "Monitoring backward" to the first data batch.	Display again.
PQMS2-ACS-INT-TC010	S7	S9-S4	At the first batch data, click the toolbar item "Monitoring backward".	Prompt "Returned the first batch data", and shut off, then return to the idle state.
PQMS2-ACS-INT-TC011	S4	S7	After idle state, click the toolbar item "Monitoring backward".	Execute monitoring backward.
PQMS2-ACS-INT-TC012	S7	S5	Click the button "Update" in the XAve-R program interface.	Display again.
PQMS2-ACS-INT-TC013	S5	S1-S4	After updating, click the toolbar item "Supervision/Tolerance setting", and enter "Drawing layout setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Display "Supervision /Tolerance setting" sheet, and shut off, then return to the idle state.
PQMS2-ACS-INT-TC014	S4	S5-S2-S4	Click the button "Update" in the XAve-R program interface. Click the toolbar item "Coefficient setting", and enter "Coefficient setting" sheet. Without input, click the button "Cancel" of the sheet.	After updating, display "Coefficient setting" sheet, and shut off, then return to the idle state.
PQMS2-ACS-INT-TC015	S4	S5-S3-S4	Click the button "Update" in the XAve-R program interface. Click the toolbar item "Drawing layout setting", and enter "Drawing layout setting" sheet. Without input, click the button "Cancel" of the sheet.	After updating, display "Drawing layout setting" sheet, and shut off, then return to the idle state.
PQMS2-ACS-INT-TC016	S4	S1-1-S4	After idle state, click the toolbar item "Supervision /Tolerance setting". In "Supervision /Tolerance setting" sheet, click the button "OK" with default correct input.	Display "Supervision/Tolerance setting" sheet. Save setting, and return to idle state.

Continued

PQMS2-ACS-INT-TC017	S4	S2-1-S4	After idle state, click the toolbar item "Coefficient setting". In "Coefficient setting" sheet, click the button "OK" with default correct input.	Display "Coefficient setting" sheet. Save setting, and return to idle state.
PQMS2-ACS-INT-TC018	S4	S3-1	After idle state, click the toolbar item "Drawing layout setting". In "Drawing layout setting" sheet, click the button "OK" with default correct input.	Display "Drawing layout setting" sheet. Save setting, and return to idle state.
PQMS2-ACS-INT-TC019	S4	S1-2-S11-S4	After idle state, click the toolbar item "Supervision /Tolerance setting". In "Supervision /Tolerance setting" sheet, click the button "OK" with null value for all inputs. Click the button "OK" of the prompt dialogue.	Display "Supervision/Tolerance setting" sheet. Prompt message of error input and shut off. Then return to idle state.
PQMS2-ACS-INT-TC020	S4	S2-2-S11-S2-2-S4	After idle state, click the toolbar item "Coefficient setting". In "Coefficient setting" sheet, click the button "OK" with null value for all inputs. Click the button "OK" of the prompt dialogue. Return "Coefficient setting" sheet, click the button "Shut off" in the up-right corner of sheet.	Display "Coefficient setting" sheet. Prompt message of error input and shut off. Return the sheet, and shut off. Then return to idle state.
PQMS2-ACS-INT-TC021	S4	S3-2-S11-S3-2-S4	After idle state, click the toolbar item "Drawing layout setting". In "Drawing layout setting" sheet, click the button "OK" with null value for all inputs. Click the button "OK" of the prompt dialogue. Return "Drawing layout setting" sheet, click the button "Shut off" in the up-right corner of sheet.	Display "Drawing layout setting" sheet. Prompt message of error input and shut off. Return the sheet, and shut off. Then return to idle state.
PQMS2-ACS-INT-TC022	S4	S6-S5	Click the toolbar item "Monitoring forward" in the XAve-R program interface. Then click the button "Update".	Execute monitoring forward, and display again.
PQMS2-ACS-INT-TC023	S5	S6	After updating, click the toolbar item "Monitoring forward".	Execute monitoring forward.
PQMS2-ACS-INT-TC024	S6	S10-S4	Click the menu item "Print-Preview" in the XAve-R program interface, and enter preview interface. Then shut off.	Preview the chart. Then shut off, return to XAve-R program interface.
PQMS2-ACS-INT-TC025	S4	S5	Click the button "Update" in the XAve-R program interface.	Display again.
PQMS2-ACS-INT-TC026	S5	S10-S4	Click the menu item "Print-Preview" in the XAve-R program interface, and enter preview interface. Then shut off.	Preview the chart. Then shut off, return to XAve-R program interface.

Continued

PQMS2-ACS-INT-TC027	S4	S7-S5-S7	Click the toolbar item "Monitoring backward" in the XAve-R program interface, and click the button "Update". After updating, click the toolbar item "Monitoring backward" again.	Execute monitoring backward. Display again. Execute monitoring backward again.
PQMS2-ACS-INT-TC028	S7	S10-S4	Click the menu item "Print-Preview" in the XAve-R program interface, and enter preview interface. Then shut off.	Preview the chart. Shut off, and return to XAve-R program interface.
PQMS2-ACS-INT-TC029	S4	S5-S10-S4	Click the button "Update" in the XAve-R program interface. Click the menu item "Print-Preview" in the XAve-R program interface, and enter preview interface. Then shut off.	Display again. Preview the chart. Shut off, and return to XAve-R program interface.
PQMS2-ACS-INT-TC030	S4	S12	Click the button "Shut off" in the up-right corner of the XAve-R program interface.	Exit program.
PQMS2-ACS-INT-TC031	S0	S6-S6-S8-S4	Enter the XAve-R program interface again, and continuously click the toolbar item "Monitoring forward" to the last data batch. At the last data batch, click the toolbar item "Monitoring forward" again. Click the button "OK" of the prompt dialogue.	Execute monitoring forward continuously. Prompt "Arrived at the last data batch". Shut off, and return to idle state.
PQMS2-ACS-INT-TC032	S4	S7-S7-S9-S4	After idle state, continuously click the toolbar item "Monitoring backward" to the first data batch. At the first data batch, click the toolbar item "Monitoring backward" again. Click the button "OK" of the prompt dialogue.	Execute monitoring forward continuously. Prompt "Returned the first data batch". Shut off, and return to idle state.
PQMS2-ACS-INT-TC033	S4	S10-S4	Click the menu item "Print-Preview" in the XAve-R program interface, and enter preview interface. Then shut off.	Preview the chart. Shut off, and return to XAve-R program interface.
PQMS2-ACS-INT-TC034	S4	S6-S1-S4	Click the toolbar item "Monitoring forward" in the XAve-R program interface. Click the toolbar item "Supervision /Tolerance setting", and enter "Supervision /Tolerance setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring forward. Then display "Supervision/Tolerance setting" sheet. Shut off, and return to idle state.

Continued

PQMS2-ACS-INT-TC035	S4	S6-S2-S4	Click the toolbar item "Monitoring forward" in the XAve-R program interface. Click the toolbar item "Coefficient setting", and enter "Coefficient setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring forward. Then display "Coefficient setting" sheet. Shut off, and return to idle state.
PQMS2-ACS-INT-TC036	S4	S6-S3-S4	Click the toolbar item "Monitoring forward" in the XAve-R program interface. Click the toolbar item "Drawing layout setting", and enter "Drawing layout setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring forward. Then display "Drawing layout setting" sheet. Shut off, and return to idle state.
PQMS2-ACS-INT-TC037	S4	S7-S1-S4	After idle state, click the toolbar item "Monitoring backward". Then click the toolbar item "Supervision/Tolerance setting", and enter "Supervision /Tolerance setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring backward. Display "Supervision/Tolerance setting" sheet. Shut off, and return to idle state.
PQMS2-ACS-INT-TC038	S4	S7-S2-S4	After idle state, click the toolbar item "Monitoring backward". Then click the toolbar item "Coefficient setting", and enter "Coefficient setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring backward. Display "Coefficient setting" sheet. Shut off, and return to idle state.
PQMS2-ACS-INT-TC039	S4	S7-S3-S4	After idle state, click the toolbar item "Monitoring backward". Then click the toolbar item "Drawing layout setting", and enter "Drawing layout setting" sheet. Without input, click the button "Shut off" in the up-right corner of the sheet.	Execute monitoring backward. Display "Drawing layout setting" sheet. Shut off, and return to idle state.
PQMS2-ACS-INT-TC040	S4	S6	Click the toolbar item "Monitoring forward" in the XAve-R program interface.	Execute monitoring forward.
PQMS2-ACS-INT-TC041	S6	S12	Click the button "Shut off" in the up-right corner of the XAve-R program interface.	Exit program.
PQMS2-ACS-INT-TC042	S0	S6-S7-S12	Enter the XAve-R program interface again, and click the toolbar item "Monitoring forward". Then click the toolbar item "Monitoring backward". Finally click the button "Shut off" in the up-right corner of the main program interface .	Start program, execute monitoring forward. Then monitoring backward. Exit program finally.

Continued

PQMS2-ACS-INT-TC043	S0	S5-S5-S12	Enter the XAve-R program interface again, and click the toolbar item "Update", and click it again. Finally Click the button "Shut off" in the up-right corner of the main program interface.	Start program, display again. Then execute update again. Exit program finally.

Table A2. State test suite of invalid one step transformation for XAve-R chart.

ID	Start state	End state of unpermitted transforming
PQMS2-ACS-INT-TC101	S_1	S_1, S_2, S_3, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}
PQMS2-ACS-INT-TC102	S_2	S_1, S_2, S_3, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}
PQMS2-ACS-INT-TC103	S_3	S_1, S_2, S_3, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}
PQMS2-ACS-INT-TC104	S_4	S_4, S_8, S_9, S_{12}
PQMS2-ACS-INT-TC105	S_5	S_4, S_8, S_9, S_{12}
PQMS2-ACS-INT-TC106	S_6	S_9, S_{10}, S_{12}
PQMS2-ACS-INT-TC107	S_7	S_8, S_{10}, S_{12}
PQMS2-ACS-INT-TC108	S_8	S_1, S_2, S_3, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}
PQMS2-ACS-INT-TC109	S_9	S_1, S_2, S_3, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}
PQMS2-ACS-INT-TC110	S_{10}	S_1, S_2, S_3, S_5, S_8, S_9, S_{10}, S_{11}, S_{12}
PQMS2-ACS-INT-TC111	S_{11}	S_1, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}
PQMS2-ACS-INT-TC112	S_{12}	S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}