논문 2022-1-2 http://dx.doi.org/10.29056/jsav.2022.06.02

An Integration Test Scenario-Based Approach to Appraise the Completeness of ERP Systems

Yukyong Kim*†

Abstract

From a qualitative point of view, the completion of software can be said to be a state in which not only functional requirements but also non-functional requirements are satisfied. Even if the developed software satisfies all the functional requirements, if the non-functional requirements are not satisfied, the degree of completeness does not reach 100%. In particular, in the case of establishing a system that integrates management resources across all areas of the business process into one, such as the ERP system, it is very important to evaluate not only the unit function but also the overall business process integration. In this case, it is necessary to judge the level of completion from an integrated point of view. Scenario-based test case design can find functional flaws for requirements, so it can be applied to various fields such as automobile international standards, railways, and national defense. In this paper, we will discuss an approach that can derive test cases and calculate the completeness by applying a scenario-based technique for software completeness assessment. This paper presents a method to apply a scenario-based test technique to software completeness appraisal on enterprise-wide business processes such as ERP. The characteristics of the ERP system will be reviewed, and the applicability of the scenario-based assessment method and its limitations will be discussed.

keywords : software testing, scenario, usecase, completeness, ERP systems

1. Introduction

Software completeness appraisal is to determine functional problems, reliability problems of processing results performed in unit tasks, operational problems, etc. for the software system developed by consignment. It is performed by analyzing various documents such as target software, proposal, development contract, system blueprint, and development work specification, and at the same time verifying whether it operates normally on the actual system [1]. From a qualitative point of view, the completion of software can be said to be a state in which not only functional requirements but also non-functional requirements are satisfied. Even if the developed software satisfies all the functional requirements, if the non-functional requirements are not satisfied, the degree of completeness does not reach 100%.

* Division of Basic Engineering, Sookmyung Women's University
† Corresponding author: Yukyong Kim (email: ykim.be@sookmyung.ac.kr)
Submitted: 2022.05.22. Accepted: 2022.06.08. Confirmed: 2022.06.20.

In the process of software development, functional and non-functional requirements should be included in the requirements specification prepared by agreement between the owner and the developer. Even if the requirements are vaguely described and there is a difference of opinion between the client and the developer, it is possible to evaluate through related data, similar cases, and expert knowledge of the appraiser, but objective grounds for the requirements must be presented [2]. In particular, in the case of establishing a system that integrates management resources across all areas of the business process into one, such as the ERP system, it is very important to evaluate not only the unit function but also the overall business process integration. In this case, it is necessary to judge the level of completion from an integrated point of view.

In this paper, we propose a method to apply a scenario-based test technique to software completeness appraisal on enterprise-wide business processes such as ERP. The characteristics of the ERP system will be reviewed, and the applicability of the scenario-based test method and its limitations will be discussed.

The structure of this paper is as follows. First, Section 2 describes the characteristics of the ERP system and the scenario-based testing technique, and Section 3 describes the scenario-based test case design method. Section 4 presents an approach for applying the test scenario to the appraisal of software completeness. A conclusion is drawn in Section 5.

2. Related Work

2.1 Characteristics of the ERP system

ERP systems inevitably accompany the innovation of the business process as it improves the current process according to the advanced process. Therefore, a different approach from the existing information systems is required to appraise ERP software. In the business aspect, ERP has a single point of failure problem in which one failure leads to the failure of the entire system because all data and transaction processing of an organization exist within one application. Therefore, it is necessary to judge the completeness from an integrated point of view [3][4].

As a generic functionality, completeness is a characteristic that becomes unrealistic if pushed to the extreme. A generic system that would work for all types of companies and industries are actually very difficult, if not impossible, to design. Depending on the nature of their physical flows, manufacturing companies may have ERP requirements. The type of ERP system can be developed taking into account the specifics of the organizations and industries that actually adopt it. As ERP systems add or eliminate specific elements for the organization, the configuration creates distinct product types and makes standard or common descriptions very difficult [5].

Since this characteristic of ERP increases the difficulty of modification, the weight for the implementation of the modification request should not be set in the same way as in the existing information systems.

2.2 Scenario-Based software testing

The international standard ISO/IEC 29119 classifies software testing design techniques into specification-based testing, structure-based testing and experience-based testing [6]. The specification-based technique designs test cases based on user requirements or specifications and models. The structure-based technique designs test cases using the logic information of the program based on the source code. The experience-based technique designs test cases based on the knowledge and experience of the person conducting the test. Specification-based testing includes equivalence partitioning, boundary value analysis, state transition testing, and scenario-based testing. Scenario-based techniques are testing based on scenarios generated by user requirements. Scenario-based testing designs test cases based on business scenarios or process flows. Scenario-based testing performed based on requirements is being applied in various fields because requirements-based testing is required not only in ISO 26262, the international standard for automobiles, but also in nuclear instrument controllers, railroads, and defense fields.

A scenario describes the functionality and behavior of software from a user-centered point of view, and is written in a way that lists the interactions between users and systems. For example, a scenario for an ATM's cash withdrawal function can be written as the following Table 1.

Table 1. Sample scenario for ATM

Scenario : Withdraw cash
Brief : This describes how a bank customer uses an ATM to withdraw money from a bank account.
Basic flow :

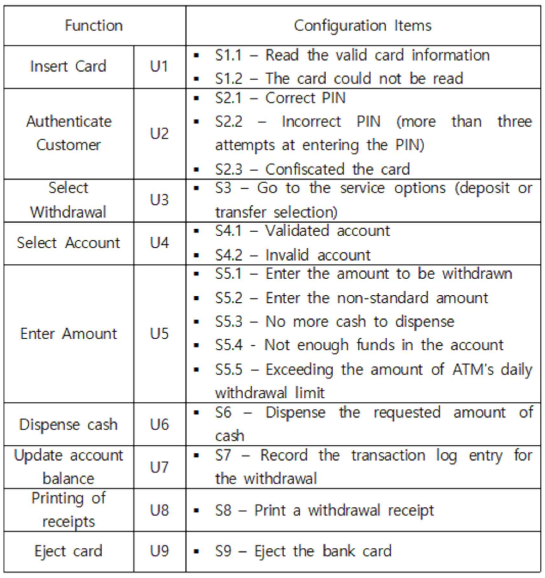
1. The customer inserts their bank card into the card reader on the ATM.
2. The system reads the bank card information from the card and check the validity of the bank card.
3. The system displays the message 'Please enter your PIN number' on the screen.
4. The customer enters their pin number.
5. The system verifies the match of PIN number with the system record.
6. The system displays the service options that are currently available on the machine.
7. The customer selects to withdraw cash.
8. The system prompts for the amount to be withdrawn by displaying the list of standard withdrawal amounts.
9. The customer enters an amount to be withdrawn.
10. The system ejects the customer's bank card.
11. The system dispenses the requested amount of cash to the customer.
12. The system records a transaction log entry for the withdrawal and returns to the initial screen.

3. Designing Scenario-Based Test Cases

Scenario-based testing shows that scenarios can be used not only to derive and document user requirements, to describe functionality, but also to validate systems during the software development process [7]. A typical form of a scenario is a use case. Use cases are intended to define interactions with users, and are the basis for the design and implementation phases, and are also used to create test cases. Because use cases describe pre- and post-conditions and performance requirements together, information necessary for testing can be provided, and practical and specific test cases can be derived. In addition to use cases, scenario-based testing can use models such as business workflow or BPMN(Business Process modelling Notation)s expressing procedures for executing business tasks. Scenario-based testing includes both normal and abnormal flows.

As a running example, we consider the use case "Withdraw cash". This use case describes how a bank customer uses an ATM to withdraw money from a bank account. Table 1 presents basic flow of events of the use case. Use case specification includes alternative flows. An alternative flow describes a use case scenario other than the basic flow that results in a user completing his or her goal. It is often considered to be an optional flow and implies that the user has chosen to take an alternative path through the system. For example, at the step 3 of basic flow in Table 1, the bank system can report that the bank card information is not valid. Then the system reports to the customer that the card could not be read and resume the basic flow at use case ends.

The scenario-based test case design process is as follows. First, the function set and configuration items are identified, and then test conditions are derived. After defining the scenario flow, design the test case. For example, the set of functions and components identified for the cash withdrawal function of an ATM are shown in Fig. 1.

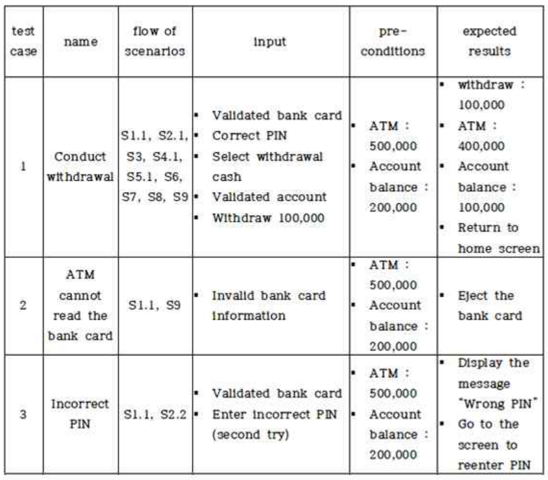


Function		Configuration Items
Insert Card	U1	▪ S1.1 – Read the valid card information ▪ S1.2 – The card could not be read
Authenticate Customer	U2	▪ S2.1 – Correct PIN ▪ S2.2 – Incorrect PIN (more than three attempts at entering the PIN) ▪ S2.3 – Confiscated the card
Select Withdrawal	U3	▪ S3 – Go to the service options (deposit or transfer selection)
Select Account	U4	▪ S4.1 – Validated account ▪ S4.2 – Invalid account
Enter Amount	U5	▪ S5.1 – Enter the amount to be withdrawn ▪ S5.2 – Enter the non-standard amount ▪ S5.3 – No more cash to dispense ▪ S5.4 - Not enough funds in the account ▪ S5.5 – Exceeding the amount of ATM's daily withdrawal limit
Dispense cash	U6	▪ S6 – Dispense the requested amount of cash
Update account balance	U7	▪ S7 – Record the transaction log entry for the withdrawal
Printing of receipts	U8	▪ S8 – Print a withdrawal receipt
Eject card	U9	▪ S9 – Eject the bank card

Fig. 1. Derived features and configuration items

Then, test conditions are derived for the defined functions and configuration items. For example, if it is a normal withdrawal scenario of cash withdrawal, the scenario flow is S1.1, S2.1, S3, S4.1, S5.1, S6, S7, S8, and S9. Several possible scenario flows can be created from test conditions, and test cases can be designed based on these derived scenario flows as shown in Fig. 2. We design eleven test cases for all scenarios in the "Withdrawal cash" use case.

Poor test scenarios reduce test quality. Therefore, it is necessary to create a test scenario based on the business process and to establish the main flow of the test scenario by the ordering company and the developer.



test case	name	flow of scenarios	input	pre-conditions	expected results
1	Conduct withdrawal	S1.1, S2.1, S3, S4.1, S5.1, S6, S7, S8, S9	• Validated bank card • Correct PIN • Select withdrawal cash • Validated account • Withdraw 100,000	• ATM : 500,000 • Account balance : 200,000	• withdraw : 100,000 • ATM : 400,000 • Account balance : 100,000 • Return to home screen
2	ATM cannot read the bank card	S1.1, S9	• Invalid bank card information	• ATM : 500,000 • Account balance : 200,000	• Eject the bank card
3	Incorrect PIN	S1.1, S2.2	• Validated bank card • Enter incorrect PIN (second try)	• ATM : 500,000 • Account balance : 200,000	• Display the message "Wrong PIN" • Go to the screen to reenter PIN

Fig. 2. Designing test cases

4. Completeness Appraisal using scenario–based testing techniques

When a test case is designed and executed based on a business scenario such as a use case, it is useful to discover functional flaws in the actual business process because the use case describes a function-oriented processing flow. In a complex system such as an ERP system, since the scenario itself is described in natural language and may not be suitable for test case design, a method of formalizing the scenario using a model such as an activity diagram or an event diagram can be used. [In [8], scenarios are derived using the UML state chart diagram.

However, it takes much more time and effort to introduce a formalized process for evaluating the completeness of software based on an enterprise-wide business process such as ERP. In this paper, instead of applying a formalized technique, we verify the test cases from the interview with the developer. It can complement the integration test scenarios.

The purpose of the completeness assessment is to verify that the system to be evaluated fully reflects the customer′s requirements and to verify the integrity of the system in the future [9]. Since the completeness evaluation is generally performed on the developed software product after the software development is completed or the product is delivered, system-level testing is required. If the scenario-based test technique is applied, the completeness can be calculated by the ratio of the passed scenarios to the entire scenario, which can be considered as a kind of functional coverage.

However, this calculation of completeness does not reflect the characteristics of the system because all scenarios and test cases are treated equally. As with the ERP system, all data and transaction processing of an organization exist within one application, and there is a high probability that a failure in one place leads to failure of the whole. In particular, the weights should not be assigned in the same way as in the existing information system to the difficulty of modification. In the ERP system, since the entire system is organically connected, the range of influence for the change increases, which increases the cost and effort.

This is because even if a low-complexity function is modified, the difficulty of the modification itself can be very high. Therefore, in the case of an ERP system, the weight scaling of a general information system may not be suitable for an integrated completeness determination.

In order to calculate the weight, the complexity of the change should be considered. In the case of the ERP system, one unit task can be linked with other systems in a complex way, so weights can be assigned according to the range of influence.

One possible option is to use a requirement system diagram. Fig. 3 is an example of a requirement system diagram. By using this, weights are separately applied according to the level of the main function, and for changes with a large impact range, additional weights can be given according to the number of unit tasks within the scope of the change.

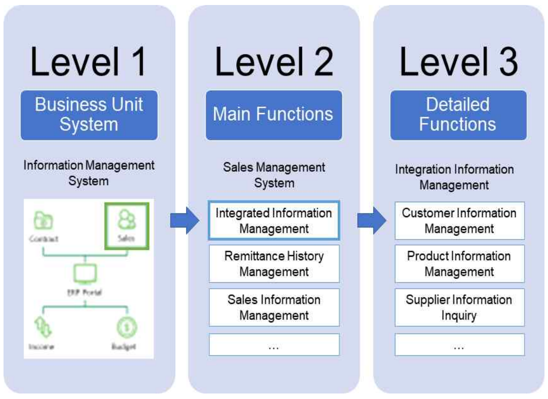

Level 1
Business Unit System
Information Management System
Contract
Sales
ERP Portal
Budget
Level 2
Main Functions
Sales Management System
Integrated Information Management
Remittance History Management
Sales Information Management
...
Level 3
Detailed Functions
Integration Information Management
Customer Information Management
Product Information Management
Supplier Information Inquiry
...

Fig. 3. An example of requirement system diagram

5. Conclusions

The scenario-based test technique is one method of performing a test by designing a test case to verify the software function using a scenario. Scenario-based test case design can find functional flaws for requirements, so it can be applied to various fields such as automobile international standards, railways, and national defense. In this paper, we discussed an approach that can derive test cases and calculate the completeness by applying a scenario-based test technique for software completeness appraisal.

References

[1] S. Hong, S. H. Park, M. S. Jeon, Y. H. Kim, J. Bae, "A Study of Software completeness appraisal and development cost calculation analysis", Proceedings of KMIS International Conference, pp.426-432, 2017.

[2] K-T Kwon, "Overview of the Appraisal of Completeness for Software", Journal of Software Assessment and Valuation, Vol. 11, No. 2, pp. 1-8, 2015. available on : http://i3.or.kr/html/paper/2015-2/(1)2015-2.pdf

[3] S. G. Yi, "An Analysis of the Importance of the Success Factors in Operation Stage of ERP System", Journal Of Service Research and Studies, Vol. 6, No. 4, pp. 35-45, 2016. DOI : https://doi.org/10.18807/jsrs.2016.6.4.035

[4] J. S. AlGhamdi, and Z. Muzaffar, "Metric suite for assuring the quality of ERP implementation and development",

Proceedings of the International Conference on Advanced Communications Technology, pp. 1348-1352, 2011. https://ieeexplore.ieee.org/document/5746054

[5] S. Uwizeyemungu, L. Raymond, "Essential characteristics of an ERP system : conceptualization and operationalization", Journal of Information and Organizational Sciences, Vol. 29, No. 2, pp.69-81, 2021. URI: https://hrcak.srce.hr/file/116377

[6] T. H. Im, "Software Testing International Standard Status(ISO/IEC/IEEE 29119)", TTA Journal, Vol. 167, pp. 96-101, 2016. https://www.tta.or.kr/data/androReport/ttaJnal/167-5-3.pdf

[7] K. Kakimoto, H. Umeda, K. Sogawa, and Y. Ueda, "A Scenario-based Approach; Assuring Effect of Software Product", Procedia Computer Science, Vol. 126, pp. 646-655, 2018. DOI : https://doi.org/10.1016/j.procs.2018.07.299

[8] Y. Lee, J. Lee, N. Kim, D. H. Lee, H. In, " A Elicitation Method an Integration Test Scenario Using Test Design Technique", Proceedings of the Korean Information Science Society Conference, Vol. 39, No. 1(B), pp. 229-231, 2012. 1598-5164(pISSN)

[9] K-T Kwon, "Software of Completeness based on Testing Technique", Journal of Software Assessment and Valuation, Vol. 6, No. 2, pp. 25-33, 2010. http://i3.or.kr/html/paper/2010-2/(3)2010-2.pdf

Authors

Yukyong Kim

2001. Ph.D. degree in Computer Science from Sookmyung Women's University
2005-2006. Postdoc. in Dept. of Computer Science at UC Davis
2006-2015. Research professor, Dept. of Computer Science and Engineering at Hanyang University, ERICA
2018-present. Professor, Division of Basic Engineering at Sookmyung Women's University
<Research interests> Software Quality Metrics, Quality of Services, Trust Evaluation for IoT services, Cloud-native applications, and Web services.