Software Project Level Estimation Model Framework based on Bayesian Belief Networks

Hao Wang
Siemens Ltd. China
CT SE
Beijing, China
wanghao@siemens.com

Fei Peng
Siemens Ltd. China
CT SE
Beijing, China
Fei.Peng@siemens.com

Chao Zhang
Siemens Ltd. China
CT SE
Beijing, China
zhaochao@siemens.com

Andrej Pietschker
Siemens AG
CT SE 1
Munich, Germany
Andrej.Pietschker@siemens.com

Abstract

Software estimation models should support managerial decision making in software projects. We experience that most of current models do not achieve this goal to the extend managers are looking for. This paper presents a software project level estimation model framework based on Bayesian Belief Networks. The framework is constructed by using four basic BBN sub-models, component estimation, test effectiveness estimation, residual defect estimation and test estimation sub-models. The integration of these sub-models achieves an estimation model suitable for project levels. With this project level estimation model, the estimation and analysis of quality, effort, schedule and scope can be carried out at both project level and specific project phase level. We show how this approach is used in a sample project, allowing project manager to implement an initial estimation, trade-off quality, effort, schedule and scope, and refine the estimation in the later phase of the project.

1 Introduction

In software project estimation and planning, it is important to balance the relationships between effort, schedule and quality, which form the three essential aspects of the famous "magic" triangle. It is widely accepted that simply estimating one of these aspects without considering the others will result in unrealistic estimations. The diversity of factors and their quantitative contribution to these three aspects are still under investigation [2]. Furthermore how to use refined information for next phase estimation of an evolving project is another challenge for decision makers.

Classical estimation models are established based on linear or non-linear regression analysis, which incorporate fixed input factors and fixed outputs. The size of the project determining the scope is modeled as a main input of such models. One representative of such models is COCOMO [2].

The most critical problem in such an approach is the wealth of data that is needed to get regression parameters, which is often impossible to get in needed quantities. It really limits their usage for project estimation. In recent years, a flexible and competitive method, Bayesian Belief Network (BBN), has been proposed for software project estimation [6, 3].

A Bayesian Belief Network is a directed graph in which the nodes represent probabilities, while the arrows between the nodes represent dependencies. It becomes attractive due to its capability of modeling uncertainty effectively, data learning and probabilistic inference. So, Bayesian Belief Network can be used to reduce the modeling data and incorporate uncertainty by introducing expert knowledge. Also, its graphic representation and reasoning methodology provide more flexibility to not only for estimation but also for trade-off analysis.

Currently studies regarding the use of BBNs in software engineering area are related to different aspects, including productivity prediction [6], quality estimation [4], and resource allocation [3]. Norman Fenton et.al developed in [3] a resource model, which can be used for trade-off analysis by considering all the factors in the "magic" triangle. How-

ever this generic approach at high level does not consider development and test phases in a project.

In this paper, we present a project level estimation model framework, which is based on four basic BBN models. The content is organized as follows. Section 2 introduces the basic sub-models, which are our foundations of the project level estimation model framework. Section 3 explains the details about how we built a project level estimation model by using the basic sub-models. In Section 4, we illustrate a model application for a sample project.

2 Basic Estimation Models for Development and Testing

In our research, we constructed four sub-models based on BBN, which are used to model the relationships between quality, effort, schedule and scope for software development and testing separately. The sub-models are:

- Component Estimation Model
  Contains variables related to component development activities which cover defect introduction, development effort, duration estimation and resources allocation.
- Test Effectiveness Estimation Model
  Contains variables related to test activities which mainly cover test effectiveness estimation.
- Residual Defect Estimation Model
  Contains variables which are used to estimate the number of residual defects after defect removal activities have taken place.
- Test Estimation Model
  Contains variables related to test effort, schedule estimation and resources allocation.

These sub-models work as basic building blocks for constructing the project level model. By using them, managers can model different projects with high flexibility. An example can be found in Section 3.

2.1 Component Estimation Model

More and more software projects are adopting component development and therefore integrating components into subsystem or products. In a large project a software system may consist of a number of components, which are designed and implemented by different development teams, or even different organizations or software vendors. Therefore, component development can be regarded as a main part in the development phase. And it is also highly related to defect introduction. Here a component refers to a software entity developed by an identifiable team and with traceability of defects found during and after its development. In our previous research, we proposed a sub-model for component estimation. Here, we simplify the structure by hiding several internal nodes, as shown in Figure 1.

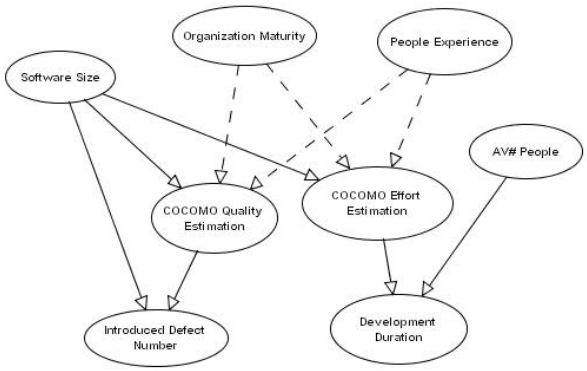

Figure 1. Component Estimation Model

In this sub-model, the number of introduced defects from development activities is calculated by multiplying component Software Size with COCOMO Quality Estimation, which is measured as defect density. The main aspects that determine the introduced number of defects are Software Size, Organization Maturity, and People Experience. The CPDs in node COCOMO Quality Estimation and COCOMO Effort Estimation are retrieved by approximating COQUALMO and COCOMOII [2] calculations using historical data from older projects.

In real projects, the resources are not distributed equally through the project duration. Different project phases may involve different staff. So, normally an average number of people can be used to depict such resource usage for the whole duration. In [5] Jensen et al. model the relationship of size, effort and schedule as in Figure 2.

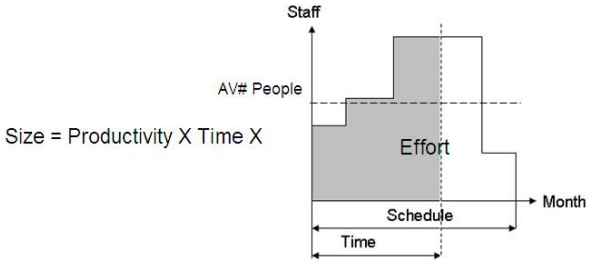

Figure 2. Relationship between Size, Effort, and Schedule

The relationship between test effort, test schedule and

average number of people can be depicted by this formula:

$$Schedule = Effort/AV\#People \quad (1)$$

By adopting such an approach, like in Figure 2, we introduced two nodes: AV# People, and Development Duration. The relationship between them is quantitatively modeled as Conditional Probability Tables (CPT) in the Development Duration node. This approach is also used in the Test Estimation sub-model, which will be introduced later. By using this sub-model, we can estimate the number of defects introduced during component development and effort and schedule needed for development.

2.2 Test Effectiveness Estimation Model

In software development, test can be regarded as the most important defect removal activity after component development. Test effectiveness, the percentage of faults that were exposed through test execution, is the most important metric for handling the number of defects. Here, a general test effectiveness estimation model is created, see Figure 3.

Figure 3. Test Effectiveness Estimation Model

We think the overall structure of this sub-model is applicable for testing activities like integration test and system test. Therefore, with updated CPTs, we could also use them to model defect removal activities in a unified way.

2.3 Residual Defect Estimation Model

Using the number of introduced defects and the test effectiveness, a simple calculation results in the number of residual defects. Such a calculation can be embedded in the Residual Defect Estimation Model, see Figure 4.

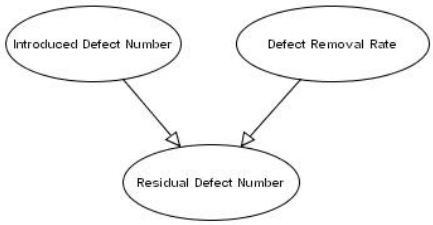

Figure 4. Residual Defect Estimation Model

2.4 Test Estimation Model

The aim of the Test Estimation Model is to estimate the effort and duration for the indicated test. Normally for a test organization or a team, with historical data, we can get the test design effort and test execution effort from historical data, which both are applied for one test case. Test case density can also be estimated using historical data through dividing the total number of test cases by the number of requirements or features under test.

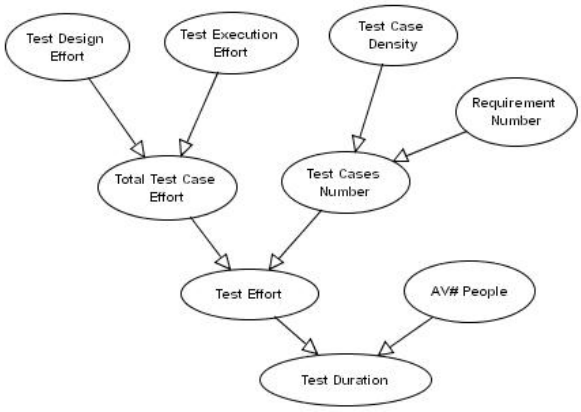

Figure 5. Test Estimation Model

So, the inputs for estimation of test effort and test duration are number of requirements and average number of people who are focusing on testing. With this model, managers can estimate the test effort and duration with the indicated test scope, which is the number of requirements here. Also a trade-off analysis for test resources can also be made using this sub-model.

3 Project Level Estimation Model Framework based on BBNs

In this section, we will introduce the project level estimation model framework, which is based on above basic sub-models. To achieve this, we simply adopt waterfall life

cycle model and use an example project to illustrate how to establish a concrete project level model using basic sub-models. Here, we just simplified software development processes by adopting waterfall model and depicted them as shown in Figure 6.

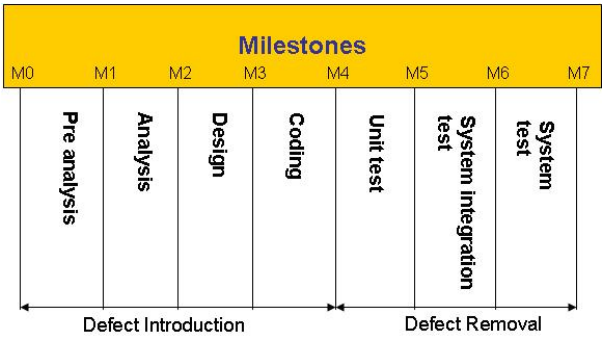

Figure 6. Simplified Process Illustration

From the Figure 6 we can see how the software development activities are aligned with different project milestones. Here we propose to take milestone M4 as the division point of defect introduction and removal activities. So our project level model should reflect such information.

Here we will use an example to explain how project are executed using these processes. Suppose there is a project aiming to develop a software system composed from three components. Each of the components is developed independently by a development team, and unit test is executed for each component separately. After that, the three components are integrated together to build the complete software system, which will then go through the integration test and system test respectively, see Figure 7.

In Figure 7, the development activities are arranged according to their finishing time. If we consider the defect flow for each activity, a pipe-and-tank model can be recognized with defect contribution and removal. Thus, all the activities are separated as two types. One type is defect introduction activities, and another is defect removal activities. The residual defects can be calculated by using the Residual Defect Estimation Model presented in Section 2.3.

Based on the structure shown in Figure 7, we can construct a project level estimation model using the basic sub-models, which are introduced in Section 2. The whole picture of the project level model for the example project is shown in Figure 8.

In this project level model, defect introduction activities are modeled using Component Estimation Model, while defect removal activities are modeled using three sub-models together. These three models are Test Effectiveness Estimation Model, Test Estimation Model and Residual Defect Estimation Model. This model was constructed using AgenaRisk [1], which is a commercial BBN tool from Agena Limited.

In this model, four phases are considered: Component Development Phase, Unit Test Phase, System Integration Test Phase and System Test Phase. Only the Component Development Phase is related to defect introduction activities, and it will be modeled using Component Estimation Model. While others are related to defect removal activities, and they will be modeled basically using Test Effectiveness Estimation Model, Test Estimation Model and Residual Defect Estimation Model. In this project level model, each rectangle represents a sub-model, and the arrow represents data flow from one model to another. For example, in the Component Development Phase, Component Estimation Model is used to model three component development quality, effort and schedule. And in the Unit Test Phase, Test Effectiveness Estimation Model, Residual Defect Estimation Model and Test Estimation Model are used to model the unit test process. The Introduced Defect Number from Component 1 Q-E-S Estimation sub-model is used as the input for the C1 Residual Defect Number sub-model.

Besides the basic sub-models introduced in Section 2, the join sub-model is used to connect phase information in the project level estimation model. In the System Integration Test phase, a join sub-model is used to combine three defect introduction outputs from Unit Test phase, as shown in Figure 9.

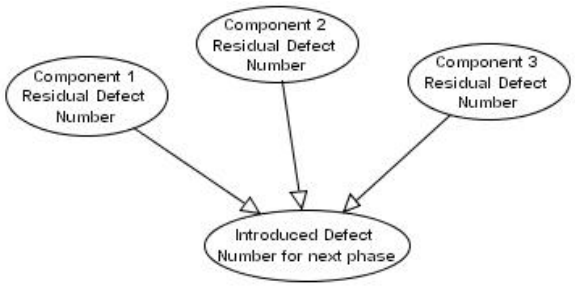

Figure 9. The joint sub-model for Entity Test phase

The CPT of Introduced Defect Number for next phase is generated by simply using addition equation. Till now, we constructed the project level estimation model for the sample project. The building method is flexible and can be used for any tailored development process by utilizing four building blocks - sub-models.

4 Project Level Estimation Model Applications

With the project level BBN model depicted in Figure 8, the relationships between quality, effort, schedule and scope

for the whole project and each phase are established. Now, project managers can get better decision support. In following sub-sections several common model applications are discussed.

4.1 Quality, Effort and Schedule Estimation

Resource estimation and allocation is critical for project planning. Here an example is used to illustrate how resource estimation can be supported by the model. In this example, we consider that managers only know that the delivered sizes of three components should be 500, 1000 and 2000 KLOC, and the average number for these three component development is 20, 30, 40. The requirement number for component 1, 2, 3 unit test, system integration test and system test is 30, 40, 50, 25 and 35 respectively. For system integration test and system test, there are on average 30 testers in each phase. This information is used as input in the Component Estimation sub-models. Thus, the model predicts that the project quality, effort and schedule can be reflected in the Table 1.

So, the total estimated effort is 26351.3 staff-days. Normally, component development and unit test are piped together, and different component development can be organized in parallel. Thus, the duration needed for component 1 development and unit test is 268.30 days. The other two is 382.56 and 101.16. So component 3 development and unit test is on the critical path of the whole project. Then the duration for the whole project is $373.10 + 32.58 + 45.81 = 451.49$ days. The quality target can be set as 13 residual defects after system test. Till now, quality, effort and schedule estimation can be done using the concrete model for the sample project.

4.2 Trade-off analysis for "Magic" triangle

Either for the whole project or for the individual phase, we can make a trade-off analysis. Such analysis can help decision makers to investigate constrains between scope, quality, effort and schedule.

Here, we will use the estimation result as our basis for trade-off analysis. In the example, we notice that the duration of component 3 development and test is much more longer than the other two. And now schedule is more important than other factors. Thus, manager wonders that if he can reduce schedule by putting more people in component 2 development and testing. Now suppose the average people for component 3 development and test will be both 50. Use the model, we can estimate the schedule for development and test is 240.63 and 39.00 days. So the duration for schedule will reduce to 358.02 days. Similarly other factors can also be changed to make trade-off analysis.

4.3 Estimation Refining

Normally with a project evolving over time, project managers can get more information, which can help them to refine the existing project estimation. Now suppose our sample project just finished entity test phase and will come into system integration phase. The manager also knows that in system integration phase, he will have 50 people with high maturity level, and the requirement number for system integration test will be reduced to 10. With all these information, the original estimation can be refined for the phases after milestone M5. The refined results are shown in Table 2.

It is clear that effort and schedule are reduced in system integration test phase, and there is no impact for system test. So, through such estimation refining, uncertainty and risks can be reduced by project managers.

5 Conclusion

We have described a project level software estimation model framework based on Bayesian Belief Networks. This model framework is based on the four basic sub-models, which are used to model quality, effort and schedule information. The integration of these sub-models then achieves the project level model for the concrete project with high flexibility. Secondly, with such project level model, project related decision makers can estimate and analyze the factors in "Magic Triangle". Furthermore, trade-off analysis can also be made in each modeled phase in the project.

Using our approach, we constructed a project level estimation model for a sample project. This project level model allows project managers to implement an initial estimation, trade-off the quality, effort, schedule and scope, and refine estimation in the later phase of the project.

Comparing to classical estimation models, the approach based on BBN can reduce the modeling data and incorporate uncertainty by introducing expert knowledge. Furthermore, the project level model is constructed by sub-models, so the estimation and analysis of quality, effort, schedule and scope can be carried out at both project level and specific project phase level.

So far we have explained our approach rather than application in real projects. Further work is related to:

- provide guidelines on how to apply the approach to specific project life cycle
- assess the validity of our approach by applying it on real projects

- introduce sub-models to reflect project activities other than development and test

References

[1] Agena Limited. AgenaRisk User Manual.

[2] B. W. Boehm. *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[3] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M. Tailor. Making resource decisions for software projects. In *Proceedings of the 26th International Conference on Software Engineering (ICSE04)*, 2004.

[4] J.-J. Gras and D. McGaw. End-to-end defect prediction. Technical report, Motorola Lab, 2000.

[5] D. R. W. Jensen, L. H. P. Sr., and W. Roetzheim. Software estimating models: Three viewpoints. *The Journal of Defense Software Engineering*, 2006.

[6] I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris. On the use of bayesian belief networks for the prediction of software productivity. *Information & Software Technology*, 45(1):51–60, 2003.

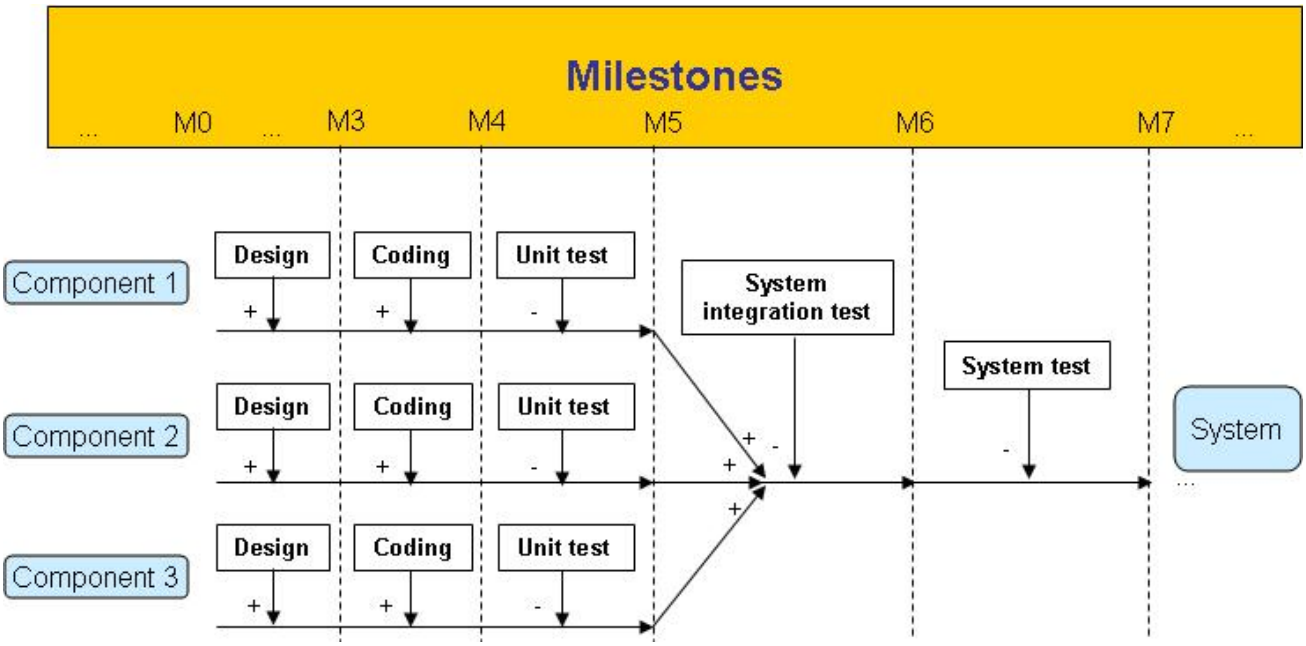

Figure 7. Example Project under Processes

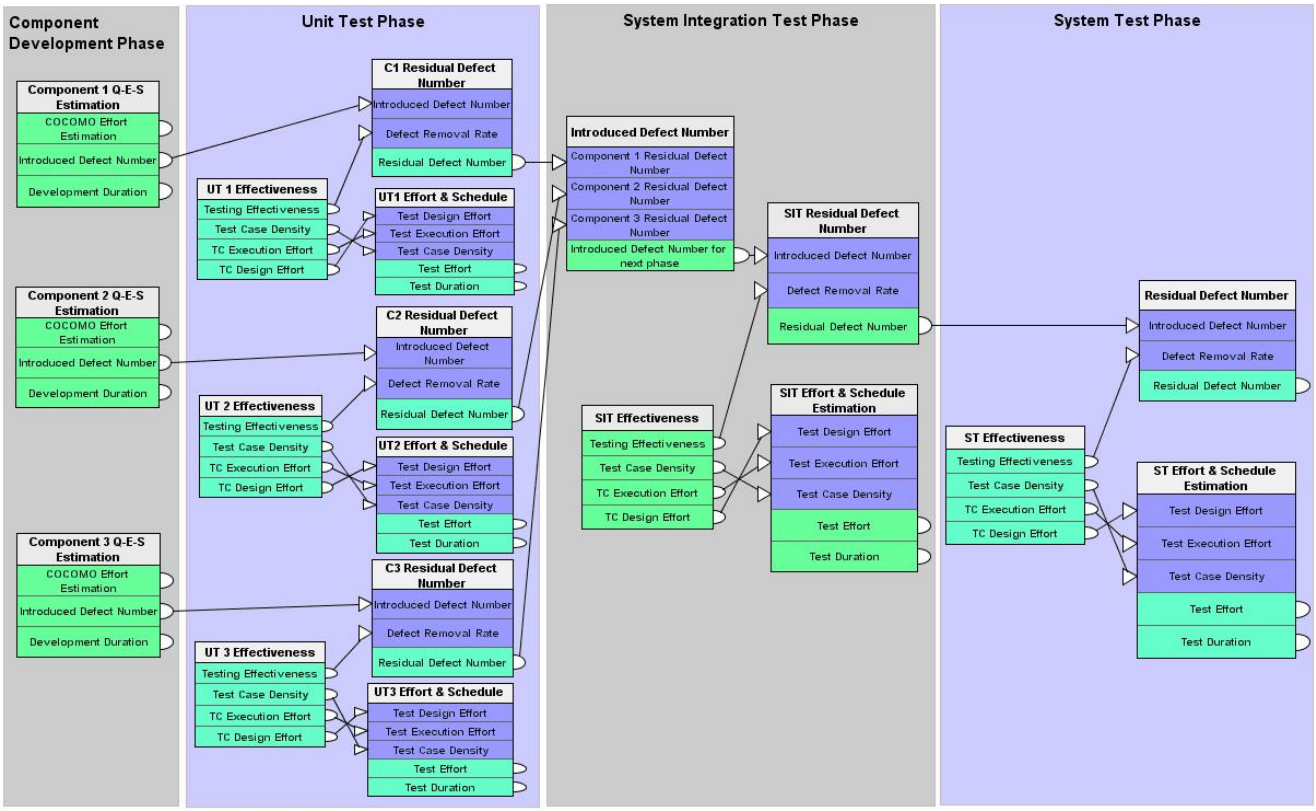

Figure 8. Project Level Estimation Model based on BBNs

Phase	Activity	Residual Defect Number	Effort (staff-day)	Schedule (day)	Schedule (AV# People)
Component Development	Component 1 Development	40	2125.00	112.05	20
	Component 2 Development	203	4375.00	156.25	30
	Component 3 Development	1050	12813.00	324.15	40
Unit Test	Component 1 Unit Test	8	1169.60	58.41	20
	Component 2 Unit Test	43	1561.90	52.21	30
	Component 3 Unit Test	220	1957.30	48.95	40
System Integration Test	System Integration Test	60	972.40	32.58	30
System Test	System Test	13	1377.10	45.81	30

Table 1. Resource Estimation Result

Phase	Activity	Residual Defect Number	Effort (staff-day)	Schedule (day)	Schedule (AV# People)
System Integration Test	System Integration Test	60	388.29	7.72	50
System Test	System Test	13	1377.10	45.81	30

Table 2. Refined Estimation Results for System Integration Test and System Test