IEICE TRANS. INF. & SYST., VOL.E99–D, NO.7 JULY 2016

PAPER

Value-Driven V-Model: From Requirements Analysis to Acceptance Testing

Youngsub HAN[†a)], ***Member***, **Dong-hyun LEE**[†], **Byoungju CHOI**[††], **Mike HINCHEY**[†††], ***and*** **Hoh Peter IN**[†b)], ***Nonmembers***

SUMMARY The goal of software testing should go beyond simply finding defects. Ultimately, testing should be focused on increasing customer satisfaction. Defects that are detected in areas of the software that the customers are especially interested in can cause more customer dissatisfaction. If these defects accumulate, they can cause the software to be shunned in the marketplace. Therefore, it is important to focus on reducing defects in areas that customers consider valuable. This article proposes a value-driven V-model (V^2 model) that deals with customer values and reflects them in the test design for increasing customer satisfaction and raising test efficiency.

key words: *V-model, test design, value-based software engineering, customer value, customer satisfaction*

1. Introduction

Customer value is an essential factor for business success in the increasingly competitive market. Woodruff defined customer value as follows [1]: *"Customer value is the customers' perception of what they want to have happen (i.e., the consequences) in a specific-use situation, with the help of a product or service offering, in order to accomplish a desired purpose or goal."* This definition of customer value shows that software companies should focus on customer values in a timely and systematic manner. According to the TechCrunch report [2], Nokia's operating loss was €1.073 billion in 2011 and more than €3 billion in 2012 because Nokia failed to cope with changes in customer values regarding the shift from feature phones to smartphones. Furthermore, Nokia had to lay off about 16,400 employees, which was 27% of its mobile and location division, during the period from Q4 2011 to 2012.

Many researchers have focused on requirements engineering in order to elicit and prioritize customer values in a systematic fashion [3]–[7]. However, as the software development process reaches the testing phase, customer values that were considered in earlier phases are often not dealt with or prioritized properly. In most projects, the test cases are assigned equal importance for every testing item. This is referred to as a 'value-neutral' approach. Boehm [8] pointed out that value-neutral testing can lead to a poor return-on-investment (ROI) and less valuable cash flows. More to the point, functions with high customer value can be late in the order of the testing process under the value-neutral testing policy. If time-to-market or budget conditions press the project schedule, these functions are often given insufficient testing. When customers find defects related to functions that customers consider important and valuable, the customers will become dissatisfied with the overall state of the software. Therefore, features that the customers view as valuable should be tested early and intensively.

In this article, we propose a value-driven V-model (V^2 model) that reflects customer values from the requirements engineering phase to the acceptance testing phase. The V^2 model advances the well-known V-model by adding an inner V-process for increasing customer satisfaction and test efficiency. The inner V-process provides a method for eliciting customer value in the requirements analysis and a method for planning customer value driven test designs for acceptance and system testing. We present the proposed model with a case study in order to explain the V^2 model specifically.

The remainder of the paper is as follows: Sect. 2 introduces the recent study of V-model, value-based software engineering, test case prioritization, and the approaches of software development. In Sect. 3, we explain the V^2 model. We introduce a method for the evaluation of the customer dissatisfaction of defects in Sect. 4. We assess the experimental results and deal with discussions in Sect. 5. In Sect. 6, we present our conclusions and describe future work.

2. Related Work

We overview related work according to the following four directions: firstly, we survey the concept of V-model and the works of advanced V-model; secondly, we take a close look at the works that aim at value-based software engineering for efficient testing; thirdly, we review the approaches of test case prioritization; and lastly, we examine the recent approaches of software development that deal with customer value.

The V-model is a well-known methodology that systems can be effectively and systematically verified and validated. During project definition phase, verification and validation are considered previously. There are several differ-

Manuscript received November 9, 2015.
Manuscript revised March 2, 2016.
Manuscript publicized April 5, 2016.
[†]The authors are with Korea University, South Korea.
[††]The author is with Ewha Womans University, South Korea.
[†††]The author is with University of Limerick, Ireland.
a) E-mail: yshan@korea.ac.kr
b) E-mail: hoh_in@korea.ac.kr (Corresponding author)
DOI: 10.1587/transinf.2015EDP7451

ent forms of the V-model: ISTQB V-model [9] throughout the testing community worldwide, Das V-Modell [10] that is the project management methodology of the German government, US V-model [11] that is the system development guide of US government, and so on. For effective testing, many researchers have advanced the V-model. The W-model [12], which is based on the V-model, clearly shows test activities in development model and clarifies the tasks of developers and testers. Mathur and Malik [13] proposed an advanced V-model for adding a quality management plan that reflects maintenance activities. Clark [14] proposes dual V-model for system of systems and V-model for family of systems. Londesbrough [15] proposes that V-model can be used for rapid application development, agile approaches, and 3rd party delivery. Gupta and Hussain [16] argued that the V-model should be used in the personal software process, the team software process, and the six-sigma process for embedded systems. However, the papers above only present the big picture for complementing the V-model and do not include practical case studies. Furthermore, there are no studies that consider customer value in a V-model setting in a systematic fashion.

For the efficient testing, value-based verification and validation (V&V) was introduced early in the 2000s and has been extended in many diverse and practical ways. Boehm proposed an agenda for value-based V&V in value-based software engineering [17]. Value-based V&V involves techniques for satisfying value objectives for software, and sequences V&V tasks in order to operate as an investment activity. Boehm also proposed the concept of value-based testing by comparing Pareto's 80:20 testing with automated test generation tool: Early tested test cases achieve about 75% to 90% the benefits of testing from an experienced report [8]. Ramler et al. [18] proposed that value-based testing consists of requirement-based testing, risk-based testing, stakeholder involvement in testing, and so on. However, these papers deal with the agenda and the concept of value-based V&V, but there is no practical case study for raising ROI. Huang and Boehm [19] suggested a value-based approach for determining when to stop testing software and release the product. Faulk et al. [38] proposed value-based software engineering that focuses on a customer's perceived value. They insisted that business requirements be decided through customer value assessment and company strategic goals, and software requirements be evaluated and prioritized considering economic perspectives. However, they did not link customer value to testing. Risk-based testing is used for optimizing available resources and time as a value-based test approach. Amland [39] proposed a risk analysis activity model for software testing. Risk exposure is calculated by multiplying the cost of fault by the probability of fault. Testing resources are focused on the areas of high risk exposure. Felderer and Ramler [40] developed a stepwise procedure for risk-based testing in industrial test processes.

Test case prioritization has been mainly used for increasing the efficiency of regression testing [41]. In recent studies, the approach of test case prioritization is applied in new testing as well as regression testing. Yoon et al. [42] proposed a method that prioritizes test cases through a correlation of requirement and risk. Srikanth et al. [43] proposed requirements-based test prioritization with customer severity, fault proneness, requirements volatility, and implementation complexity as risk factors. Li et al. [20] proposed a value-based testing method that prioritizes testing based on business importance, quality risk, and testing cost. The priority value for testing is calculated by multiplying the business importance and the quality risk and dividing by the cost of testing. However, the priority value may be affected severely by the cost of testing. The value that the customers place on a particular feature may also be affected by the cost of testing this feature. Li and Boehm [44] focused on test prioritization by risk reduction leverage that divides risk exposure by the risk reduction cost. Risk exposure is calculated using the difference between initial risk exposure and the risk exposure of risk reduction effort afterwards. Test items are ranked by the risk reduction leverage. However, we focused on test prioritization using the degree of relationship between the customer value factor and the test item. Test items are ranked by how well they represent the customer value. Higher the customer value is, higher priority of its test case is.

In last decade, the methodologies of software development have been evolved by the mainstream of agile software development and user-centered design. Agile software development relies on people and their creativity rather than on processes for overcoming the problems of traditional software methodologies [21]. User-centered design focuses on the goals and needs of the software's end-users for delivering the appropriate usability of software [22], [23]. It is important to focus on what customers consider valuable. User experience (UX) deals with total experience ranging from traditional usability to beauty, hedonic, affective or experiential aspects of technology use [24]. Customer value as well as technical attributes and consequences are important factors to raise customer satisfaction in UX. Recently many researchers have presented the integrated approach of agile software development and user-centered design [25], [26]. Brhel et al. [22] presented principles of user-centered agile software development (UCASD). They reviewed the papers of UCASD by classifying publication with the dimension of process, practices, people & social, and technology. Value-driven approach of software development is similar to the concept of user-centered design. Value-driven approach focuses on the high-level needs or system features that is prioritized by stakeholders based on the perceived business value [27], [28].

3. Value-Driven V-Model

The proposed value-driven V-model (V^2 model) revises the well-known V-model by adding an inner V-process as shown in Fig. 1. The V-model was developed for effective test design at each stage of the software development process. The V^2 model includes acceptance test design and system test

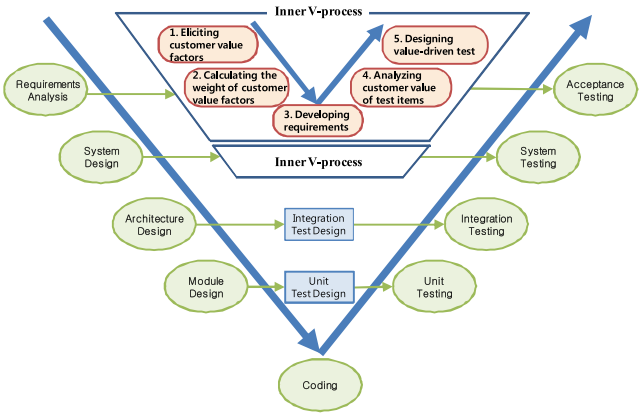


Fig. 1 Overview of V^2 model

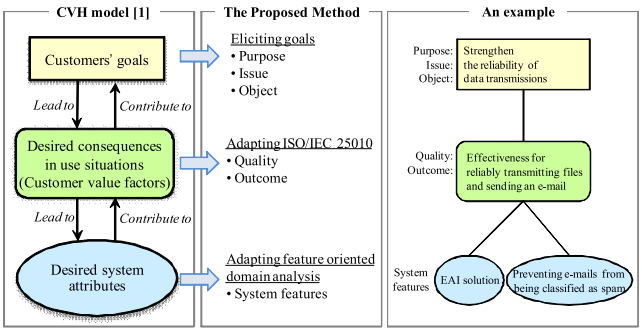


Fig. 2 Value elicitation method based on the CVH model

design. The inner V-process for acceptance test design aims to develop acceptance test cases based on customer value during the requirements analysis phase. The inner V-process includes the following five steps: eliciting customer value factors, calculating the weight of customer value factors, developing requirements, analyzing customer value of test items, and designing value-driven test. Step 1 and 2 are commonly used steps for value-driven test design. During the system test design process, steps 1 and 2 of the inner V-process are omitted, because the results from the inner V-process for acceptance testing are used instead.

3.1 Step 1: Eliciting Customer Value Factors

We define the customer value factors (CVFs) as the desired consequences when customers use software. The CVFs are used afterward to analyze the customer value of test items and to elicit important customer requirements. Before customer requirements are specifically developed, CVFs are elicited. In order to elicit CVFs, we adopted the concept of the customer value hierarchy (CVH) model [1] that is shown in Fig. 2, and applied the CVH model through focus group meeting and personal interviews in order to gather customers' requests. Customer value is elicited systematically in terms of business strategy from CVH model. The CVH model includes the following three hierarchical stages: customers' goals, desired consequences in use situation, and desired system attributes.

The customers' goals represent the desired ultimate end states that the customers expect to occur when they use the software. A goal is defined from the customer's viewpoint and is generally described by using a purpose, an issue, and an object. The purpose includes a key verb that describes what the customers want to achieve. The issue refers to an important subject area that requires a solution. The object refers to the target of the solution (i.e. a product, service, process, or resource). An example of value elicitation is presented on the right in Fig. 2. An example of goal represents the purpose as "strengthen," the issue as "the reliability," and the object as "data transmissions." When these elements are synthesized, the resulting goal is to "strengthen the reliability of data transmissions."

The desired consequences in use situations represent the required qualities for the outcomes that result from using the software. In this article, we present the desired consequences as CVFs. The CVFs are the elements for achieving the customer's goal. The CVF is described by quality and outcome. Quality related to outcome is selected in order to differentiate goals from the quality in use as defined in ISO/IEC 25010 [29]. We adopt the quality in use to reflect quality characteristics from customer's viewpoint, when customer uses the system. Outcome captures what customers want to do or the service that customers want to receive. Outcome is directly related to the accomplishment of the goal. For an example, in Fig. 2, the outcome of a CVF is "reliably transmitting files and sending an e-mail." The required quality in use for the outcome is "effectiveness." The resulting CVF is "effectiveness for reliably transmitting files and sending an e-mail."

The desired system attributes represent system features for accomplishing the CVFs. The system development team considers the latest trends, time-to-market, and costs while eliciting the main system features for accomplishing the CVFs. We adapt a feature oriented domain analysis [30] while eliciting system features. A feature is a prominent or distinctive customer-visible aspect or characteristic of a system. In Fig. 2, system features are elicited based on "effectiveness for reliably transmitting files and sending an e-mail." As a result, enterprise application integration (EAI) solution is considered in order to enhance fault tolerance and the recoverability of file transmissions. In addition, the function that prevents e-mails from being classified as spam is considered in order to create effective e-mails.

As a case study, we applied the V^2 model to the data exchange system (DES) from the Defense Agency for Technology and Quality (DTaQ). The DES is a system for raising convenience for users during data transmissions between physically disconnected networks under tightened security. The DES is required to transmit files and e-mails from the internet to an intranet or vice versa. In this case study, there were four goals, eleven CVFs, and eighteen system features as shown in Table 1.

Table 1 CVFs that were elicited based on CVH model

Customers' goals	Desired consequences (CVFs)		Desired system attributes (System features)
Improve the convenience of data transmissions	CVF_1	Effectiveness for convenient system login	Single sign on (SSO)
	CVF_2	Effectiveness for reducing workload related to sending files between disconnected networks	Manager's approval for sending e-mails, Managing reports for e-mails that have been sent, Reusing sent e-mails, Grouping e-mail list, Sending multiple files at once
	CVF_3	Efficiency for transmitting files conveniently from the internet to the intranet	Commercial off-the-shelf (COTS) for file uploads/ downloads, Sending multiple files at once
	CVF_4	Efficiency and effectiveness for sending an e-mail conveniently from the intranet to the internet	COTS for file uploads/ downloads, Writing an e-mail format for general e-mails, Alarm about approval before sending e-mails, Converting e-mail list from spreadsheet into DB
	CVF_5	Effectiveness for supporting accuracy of information in e-mails that are received	Mail reception information
Strengthen the reliability of data transmissions	CVF_6	Effectiveness for reliably transmitting files and sending an e-mail	EAI solution, Preventing e-mails from being classified as spam
	CVF_7	Trust for uploading/downloading files	COTS for file uploads/ downloads
	CVF_8	Effectiveness for the prevention of viruses	Running anti-virus Solution before sending files
Tighten the security of intranet data	CVF_9	Effectiveness for the prevention of unauthorized file leaks	Manager's approval for sending e-mails
	CVF_{10}	Effectiveness for monitoring of transmitted data	Retrieving the logs for e-mail transmissions, Converting the logs for e-mail transmissions into spreadsheet
Increase user friendliness of user interface	CVF_{11}	Comfort for user interface related to transmittals of files and e-mails	Understandability of user interface, Color design harmonized with other system

3.2 Step 2: Calculating the Weight of Customer Value Factors

In this step, the weight of CVFs is calculated to analyze the customer value of test items in Step 4. In order to calculate the weights of the CVFs, there are methods: direct subjective evaluation, SMART [31], pair-wise comparison, and a geometric progression. The pair-wise comparison method is adopted to calculate the weight of CVFs. The pair-wise comparison of CVFs compares CVFs in pairs to judge which of the CVFs is more valuable. The analytic hierarchy process (AHP) [32] is a pair-wise comparison method. Because the AHP has an advantage of verifying the consistency of the pair-wise comparison by consistency ratios, the AHP is adopted. Customer questionnaires on the pair-wise comparison of CVFs are gathered, and the weights of the CVFs are calculated. However, there is the limitation of

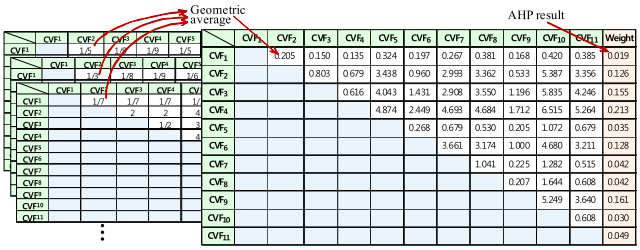


Fig. 3 Calculating the weight of CVFs by AHP: Left tables show the results of customer questionnaires. The right table presents the geometric average values of the customer questionnaires and the AHP result according to these geometric average values.

AHP. When the scale of AHP is large, it is difficult to compare pair wise. When the number of CVFs is large, clusters or pivots method [33] is useful to calculate the weight of CVFs.

In this case study, the method for calculating the weights of the CVFs is presented in Fig. 3. We selected ten customers that are members of business department. The selected customers answered questionnaires about the pair-wise comparison of CVFs. After gathering customers' questionnaires about pair-wise comparisons of CVFs, we calculate the geometric average of each pair-wise item and obtain the weight of the CVFs by applying the AHP method to the results. The weight of CVF_1 is 0.019, that of CVF_2 is 0.126, and so on. The sum of the weight of the CVFs is 1. The consistency ratio needs to be below 0.1 in order for the consistency to be valid. This case had valid consistency, because the consistency ratio from the AHP was 0.028.

3.3 Step 3: Developing Requirements

This step includes requirements analysis phase. Requirements are elicited, analyzed, specified, and validated according to the requirements engineering [34]. Customer requirements describe the functional and non-functional requirements without detailed technical knowledge [35]. The customer requirements reflect the customers' goals and CVFs during the user scenarios. System specification describes system requirements that are expanded versions of the customer requirements [35]. The system specification reflects the CVFs and system features in detail and determines how the system should be configured in order to support the customer requirements.

In this case study, we deal with the system test design. The system specification is presented in Table 2. Twenty-four system requirements were developed.

3.4 Step 4: Analyzing Customer Value of Test Items

The customer value of test item (CVTI) is the importance of test item from customer's viewpoint. Test item is a group of functions, transaction, or processes. A test item consists of a group of test cases, test scenarios, or test suites. The computation of CVTI is to calculate how valuable a test item is. In this step, the test designer calculates the importance

Table 2 Development information of system specification

No.	System requirements
SR_1	Log into the intranet system using ID/password or SSO
SR_2	Update organization information automatically
SR_3	Update users information automatically
SR_4	Download transmitted files from the internet
SR_5	Fill out e-mail contents and request transmittal of intranet data to the internet
SR_6	Approve request for sending intranet files to internet
SR_7	Set user's e-mail ID
SR_8	Manage e-mail list for e-mail group
SR_9	User's Guide for intranet system
SR_{10}	Statistics about data transmitted to the internet
SR_{11}	Statistics about data received from the internet
SR_{12}	Search data report that was transmitted to the internet
SR_{13}	Search data list that was received from the internet
SR_{14}	Configure system manager
SR_{15}	Configure file size for transmission in the intranet
SR_{16}	Monitor EAI log for data transmission
SR_{17}	Monitor logs for users logging in
SR_{18}	Log into internet system
SR_{19}	Upload data to the internet
SR_{20}	User's guide for internet system
SR_{21}	Configure file size for transmission to the internet
SR_{22}	Transmit data from the internet to the intranet using EAI
SR_{23}	Transmit data from the intranet to the internet by EAI
SR_{24}	Send e-mail using the public e-mail server

of CVTI by using the CVF and its weight from Step 1 and 2. The process of calculation of CVTI is as follow: Firstly, calculate the degree of correlation between a test item and a CVF. Secondly, multiply the degree of the correlation by the weight of the CVF. Lastly, summarize the second value of each CVF on the test item. When a test item has high relationship with CVFs having high weight, the test item is important from customer's viewpoint. In order to calculate the correlation effectively, we employ a decision matrix that is adapted from the VIRE model [3]. A CVTI is expressed as follows:

$$\begin{aligned} CVTI_i &= (R_{i,1} \times weight\_CVF_1) \\ &\quad + (R_{i,2} \times weight\_CVF_2) + \ldots\ldots + \\ &\quad (R_{i,j} \times weight\_CVF_j) \\ &= \sum_{j=1}^{n} (R_{i,j} \times weight\_CVF_j) \end{aligned}$$

where, $R_{i,j}$ is the degree of correlation for a test item i (TI_i) and a customer value factor j (CVF_j). The correlation ($R_{i,j}$) is assessed on a scale using the following symbols: ● (9: strong relationship), ◉ (3: moderate relationship), and ○ (1: weak relationship). The $weight\_CVF_j$ is the weighted value of the CVF_j in Step 2. The CVTI is high when test items and CVFs are strongly related, and vice versa.

In this case study, test items are classified according to the system requirements. The value of a CVTI and its priority in the system requirements shows on the right of Fig. 4. SR_6 is estimated as the highest priority CVTI.

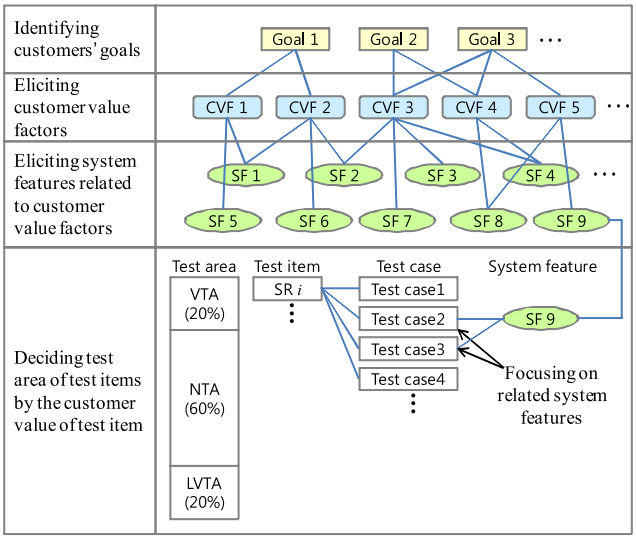
	CVF_1 (0.019)	CVF_2 (0.126)	CVF_3 (0.155)	CVF_4 (0.213)	CVF_5 (0.035)	CVF_6 (0.128)	CVF_7 (0.042)	CVF_8 (0.042)	CVF_9 (0.161)	CVF_{10} (0.030)	CVF_{11} (0.049)	CVTI	Priority
SR_1	●										◉	0.318	21
SR_2	◉											0.057	24
SR_3	●											0.171	23
SR_4		●	●				●				●	3.348	5
SR_5		●		●			●		◉		●	4.353	2
SR_6		●		●					●		●	4.941	1
SR_7		◉		◉							●	1.458	9
SR_8		●		●							●	3.492	4
SR_9											●	0.441	17
SR_{10}											●	0.441	17
SR_{11}											●	0.441	17
SR_{12}				◉	●				●	●	●	3.114	6
SR_{13}			◉							●	●	1.176	10
SR_{14}									○		●	0.602	14
SR_{15}		○									●	0.567	15
SR_{16}						◉					●	0.825	13
SR_{17}									◉		●	0.924	12
SR_{18}	●										◉	0.318	21
SR_{19}		●	●				●	●			●	3.726	3
SR_{20}											●	0.441	17
SR_{21}		○									●	0.567	15
SR_{22}			●			●						2.547	8
SR_{23}				●		●						3.069	7
SR_{24}						●						1.152	11

Fig. 4 Analyzing CVTIs in the data exchange system

3.5 Step 5: Designing Value-Driven Test

The test designer designs test based on customer requirements for acceptance tests and based on system requirements for system tests in this step. The proposed test design strategy is that the test designer concentrates on valuable test items with the viewpoint of customers. It is important to focus on reducing defects in areas that customers consider valuable. Further, the test items are prioritized according to CVTI. Then, the test area of the test items is decided by the priority of the test item. The development of the test cases differs according to the test area. The test designer generates exhaustive test cases for test items with high priority CVTI, but reduces test cases for test items that have low priority CVTI. This ensures that the system features that are related to CVFs are focused on and are fully tested. However, there are limitations due to time and the cost of testing. It takes much time to generate test cases from every system feature related to the CVFs. It requires much expense to test all test cases that are generated.

As a result, we propose that the testing effort should be divided into a valuable test area (VTA), a normal test area (NTA), and a less valuable test area (LVTA) based on the CVTI. According to Pressman and Maxim [36], 20% of the code has 80% of the errors. It is important to find out 20% of the code that has many errors. Therefore, to find more defects on valuable area from customer point of view, we focus on high CVTI. Test cases are enlarged in 20% of high CVTI, but reduced in 20% of low CVTI, because planned testing cost should be used efficiently. The VTA covers the test items that are in the top 20% based on the priorities that are derived from the CVTI. The NTA covers the test items that are in the next 60% of the test items based on priority and the LVTA covers the final 20% of the test items. For the VTA, the test designer generates more 20% test cases related to system features of CVF than NTA. For the NTA, the



* SF: System feature, SR: System requirement

Fig. 5 Mapping the relation among customer goals, CVFs, system features, test items, and test cases.

test cases are developed from the main flow and the alternate flow of requirements as in traditional value-neutral testing. For the LVTA, less 20% test cases are developed than NTA. The test designer only develops test cases for the main flow of requirements in order to reduce the number of test cases. The VTA and LVTA can be enlarged or reduced by comparing the planned testing cost during project planning with the estimated testing cost during test design. When the estimated testing costs are lower than the planned testing costs, the VTA can be enlarged. When the estimated testing costs are higher than the planned testing costs, the LVTA can be enlarged in order to reduce the estimated testing costs.

Figure 5 presents the example of mapping the relation among customer goals, CVFs, system features, system requirements, test items, and test cases.

For this case study, the VTA, NTA, and LVTA were determined as follows. The requirements that fell into the 20% sector with high priorities (i.e., SR_6, SR_5, SR_{19}, and SR_8) were assigned to the VTA. The requirements that fell into the 20% sector with low priorities (i.e., SR_2, SR_3, SR_1, and SR_{18}) were assigned to the LVTA. The rest of the requirements were assigned to the NTA. Test cases were generated differently for the different areas based on the test design strategy. For SR_6, more test cases were developed for checking the manager approvals of e-mails in order to ensure the security requirements related to CVF_9. Test cases were also added for CVF_4 in order to verify that the contents of alarm messages were appropriate and to make sure that alarm messages arrived in a timely fashion after managers' approvals. Table 3 presents a test case of SR_6. Additional test data and attachments with many different files sizes were used in order to increase the number of test cases for SR_5 and SR_{19} beyond the number of test cases for traditional value-neutral testing. On the other hand, the test cases for SR_2, SR_3, SR_1, and SR_{18} were reduced by limiting the amount of test data.

Table 3 A test case (checking alarm message) of SR_6

Step no.	Action (including input data)	Expected results	Remarks
1	Click on the "Approval of sending e-mail" menu.	Approval of sending e-mail is displayed.	
2	Retrieve the sending e-mail with the conditions of date, approval status, writer, receiver, and title.	According to the conditions, the results are displayed.	
3	Select an e-mail to be approved.	The selected e-mail is displayed.	
4	Click on the "Approval" button.	The alarm message (e-mail) of approval arrives to the e-mail writer within 5 seconds of the manager's approval.	Add response time.
5	Log in to Mail System with writer's id	The main page of Mail System is displayed.	
6	Reads the alarm message of the manager's approval.	The e-mail content includes the e-mail title, approval date, and link for viewing the approval status.	
7	Click "link" for viewing the approval status.	Pops up a new window. The e-mail list to send with the approval status is displayed.	

4. Evaluation Method

The reason for considering customer value is to increase customer satisfaction through efficient tests. In order to assess customer satisfaction after the proposed method was applied, we calculated the degree of customer dissatisfaction (DCD) related to defects. We pay our attention that finding and eliminating the defects of high DCD raise customer satisfaction. We propose that DCD is calculated by multiplying customer severity by defect severity. Customer severity indicates the degree to which a customer considers a defect to be severe. Defect severity is a classification of defects to indicate the degree of negative impact on the quality of software. The test team or consultants who know the V^2 approach calculate customer severity by analyzing the relationship between a defect and CVFs. The test team assigns the defect severity. The formula is as follows:

$$DCD = Customer\ Severity \times Defect\ Severity$$

$$DCD_i = \left(\sum_{j=1}^{x}(R_{i,j} \times weight\_CVF_j) + 1\right) \times (Defect\ Severity)_i$$

where, $R_{i,j}$ is the degree of correlation for $Defect_i$ and CVF_j, $weight\_CVF_j$ is the weighted value of the CVF that was calculated in Step 2. The correlation ($R_{i,j}$) is assessed on a scale using the following symbols: ● (9: strong relationship), ◉ (3: moderate relationship), and ○ (1: weak relationship). The defect severity is set using the following values: trivial (1), minor (2), major (3), and critical (4). If a defect is not related to the CVFs and there is no 'add 1' in the formula, then the customer severity would be 0 regardless of defect severity. In order to avoid this situation, we set 1 as the minimum value of customer severity.

Figure 6 presents the calculation matrix of DCD in project 1. The value of right parenthesis of CVF refers to the weight of CVF in Step 2. For example, $Defect_1$ is moder-

	$Defect_1$	$Defect_2$	$Defect_3$	$Defect_4$	$Defect_5$	$Defect_6$	$Defect_7$	...
CVF_1 (0.019)								
CVF_2 (0.126)			●					
CVF_3 (0.155)					●	●	●	
CVF_4 (0.213)			●		●	●		
CVF_5 (0.035)								
CVF_6 (0.128)								
CVF_7 (0.042)					⊙	●		
CVF_8 (0.042)								
CVF_9 (0.161)								
CVF_{10} (0.030)			⊙					
CVF_{11} (0.049)	⊙					⊙	●	
Customer Severity	1.147	1.000	4.141	1.000	4.438	4.837	2.836	
Defect Severity	1	2	2	1	2	1	2	
DCD	1.147	2.000	8.282	1.000	8.876	4.837	5.672	

Fig. 6 Calculation matrix of DCD in Project 1

ately related to CVF_{11}, therefore customer severity is 1.147. $Defect_1$ is trivial (1) in the defect severity. The DCD of $Defect_1$ is 1.147 by multiplying customer severity by defect severity.

5. Results and Discussion

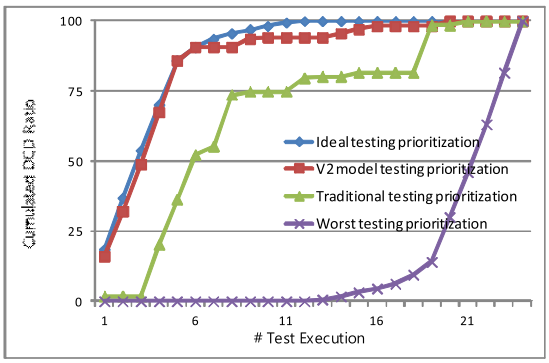
We applied the V^2 model to three projects in Defense Agency for Technology and Quality in Korea. Table 4 shows the information of project development. Project 1 is a data exchange system as we use a case study in Sect. 3. Project period was four months. The users are over six hundreds. It is developed in ASP and C#. Project 2 is a price information management system for defense acquisition. Project period was seven months. The users are over one hundred. It is developed in Java, JSP, and Flex. Project 3 is an assessment management system for defense research. Project period is eight months. The users are over one thousand. It is developed in Java and JSP.

Figure 7 presents the accumulation ratio of DCD on defects. The graph for ideal testing prioritization accumulates DCD in order of the more amounts of DCD on detected defects. The graph for the V^2 model testing prioritization accumulates DCD in order based on the prioritizations of test items by the V^2 model. The graph of traditional testing prioritization accumulates defects in order based on business flow of test items. The graph of worst testing prioritization accumulates DCD in order of the less amounts of DCD on detected defects. The graph of the V^2 model is adjacent to ideal testing prioritization. The VTA, about 20% of all test items, accounted for 67.4% of the total DCD in project 1, 59.2% in project 2, and 69.2% in project 3. We found out that V^2 model detects well defects which contain high DCD in the VTA. When there is insufficient time to test, the V^2 model increases customer satisfaction by early finding out defects related to CVFs and complementing them.

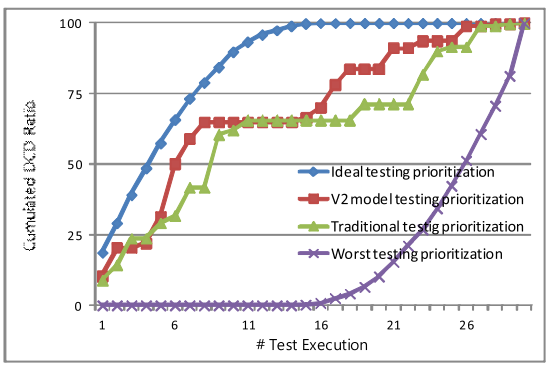
It is important to consider ROI during testing, because there often is not enough time to support proper testing. As a result, test designers look for ways to obtain better results from the same level of investment. Table 5 shows the

Table 4 Information of project development

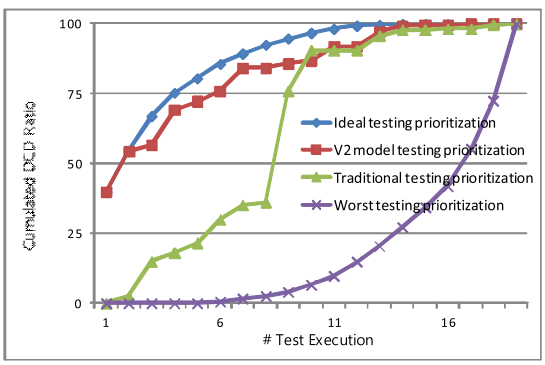
	Project 1	Project 2	Project 3
Number of CVFs	11	11	9
Number of system features	18	21	14
Number of system requirements	24	30	26
Number of test cases	120	225	127
Development period	4 months	7 months	8 months
Development language(s)	ASP, C#	Java, JSP, Flex	Java, JSP
Number of lines of code	175,773	1,955,333	610,984



(a) Project 1



(b) Project 2



(c) Project 3

Fig. 7 Accumulation ratio of DCD

Table 5 The number of detected defects by applying the V^2 model

Project No.	Total	VTA	NTA	LVTA
Project 1	49	36	11	2
Project 2	146	74	59	13
Project 3	196	109	85	2

number of detected defects on the above mentioned three projects. More defects were detected in the VTA than in other test areas. The VTA accounted for 73.5% of the total defects in project 1, 50.7% in project 2, and 55.6% in project 3. The requirements where customers put more value have more complex logic and many functions. In addition, they are frequently-used functions. Programmers tend to make mistakes during programming requirements where the customer put more value due to the corresponding complexity. When there is insufficient time for testing, it is more effective for the tester to test the VTA first from the view point of finding out defects.

Although we have successfully applied and evaluated the V^2 model in several case studies, but there are some threats to validity of these experiments as follows:

- *Testers' ability can affect defect detection:* We assumed that defects are detected through test cases regardless of the testers' ability, because it is tested by specified test cases. However, testers' ability actually affects defect detection, when test cases are not specified in detail.
- *CVF can conflict with other CVF:* CVF can affect other CVF, for example customer value related to efficiency can negatively influence on customer value related to security. We do not consider mutual interference in these experiments.
- *The V^2 model can cost more than traditional software development:* We assumed that there is no difference in the cost between the V^2 model and traditional software development. However, the V^2 model can require considerably more time and effort than traditional software development without appropriate tool support, because the V^2 model has more steps for creating the artifacts of CVFs, calculating the weight of CVFs, planning the tests, and designing the tests than traditional software development.
- *For the same given CVFs of a system, different test teams may end up with different test sets:* There is no formal algorithm to develop test cases. According to test teams, test cases can be developed differently.
- *We applied our proposed model to three projects. It needs to apply more projects to generalize our method.*

The results from our proposed V^2 model correspond to the results for ideal testing prioritization as depicted in Fig. 7, however there was a gap. Project 2 had a larger gap than the others, because many defects were detected in the back end of the NTA. After further analysis, we discovered that some of the requirements for the NTA were coarsely developed due to a lack of development time. We need to consider other factors in order to raise the early defect detection ratio.

We elicit CVFs and calculate the importance of CVFs from customers in the V^2 model. When customers have no the experience, interest, or knowledge of the developing software, they will not identify CVFs properly. If CVFs are not well elicited and the importance of the CVFs is not properly calculated, it will affect test design incorrectly. Then, good test results cannot be achieved. It is needed to select customers that have considerable experience of the domain and to educate the customers about the V^2 approach.

6. Conclusion

This article presents the V^2 model that reflects customer value from requirements analysis to acceptance testing. The V^2 model embodies a method for eliciting customer values that is based on the CVH marketing model. We elicit goals, CVFs and system features, and calculate customer value systematically. As a result, the value-driven test design method focuses on test items that rated as important based on customer values. It is important to achieve high quality in areas that the customers view as valuable, because customers are more dissatisfied when defects are detected in these areas of the software. In order to ensure high quality in these areas, test designers need to apply the V^2 model.

We discovered that customer value is a critical indicator that is used to raise customer satisfaction and increase test efficiency. After the application of the V^2 model, the cumulated DCD ratio of the V^2 model testing prioritization was adjacent to that of ideal testing prioritization. The V^2 model increases customer satisfaction by complementing at an early stage the detected defects that are highly related to customer value. The V^2 model will be effective to develop a system that considers customers' preferences first. The V^2 model is more efficient than traditional testing, because more defects were detected early in VTA. The V^2 model can be used to develop an application or a prototype when the time that is allocated for testing is inadequate.

For future studies, we plan to elaborate the V^2 model by applying it to additional industrial projects. To gain more validation, it needs to be applied in many industries and various domains. We will extend the V^2 model so that customer value is considered during regression testing. More studies are needed in order to find effective methods for eliciting and validating CVFs, because CVFs crucially influence the V^2 model. As big data and social networking services (SNS) have been more pervasive, it has become possible to use them to elicit the CVFs for general-purpose software. In order to select the priority of CVFs and find valuable test area, the Kano model [37] is considered to be applied during requirements engineering, because the Kano model helps to select attractive requirements and increase customer satisfaction.

Acknowledgments

This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3C4A7033345), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and

Technology (2012R1A1A2009021).

References

[1] R.B. Woodruff, "Customer Value: The Next Source for Competitive Advantage," Journal of the Academy of Marketing Science, vol.25, no.2, pp.139–153, 1997.
[2] N. Lomas, "Innovate Or Die: Nokia's Long-Drawn-Out Decline," http://techcrunch.com, Dec. 31st, 2012.
[3] S. Kim, H.P. In, J. Baik, R. Kazman, and K. Han, "VIRE: Sailing a Blue Ocean with Value Innovative Requirements," IEEE Software, vol.25, no.1, pp.80–87, Jan./Feb. 2008.
[4] N. Kukreja, "Decision theoretic requirements prioritization: A two-step approach for sliding towards value realization," Software Engineering (ICSE), pp.1465–1467, 2013.
[5] M. Harbers, C. Detweiler, and M.A. Neerincx, "Embedding Stakeholder Values in the Requirements Engineering Process," REFSQ 2015, LNCS 9013, vol.9013, pp.318–332, 2015.
[6] M. Bano and N. Ikram, "Addressing the Challenges of Alignment of Requirements and Services: A Vision for User-Centered Method," APRES 2014, CCIS, vol.432, pp.83–89, 2014.
[7] N. Kukreja, S.S. Payyavula, B. Boehm, and S. Padmanabhuni, "Value-Based Requirements Prioritization: Usage Experiences," Procedia Computer Science, vol.16, pp.806–813, 2013.
[8] B. Boehm, "Value-Based Software Engineering: Overview and Agenda," In Value-Based Software Engineering, S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Grünbacher (Eds). pp.3–14, Springer, Berlin, Heidelberg, 2006.
[9] International Software Testing Qualifications Board, "Certified Tester Foundation Level Syllabus," 2011.
[10] Bundesrepublik Deutschland, "DAS V-MODELL®," http://v-modell.iabg.de/.
[11] U.S. Department of Transportation Federal Highway Administration Federal Transit Administration, "System Engineering for Intelligent Transportation System," Jan. 2007.
[12] A. Spillner, H. Bremen, and K. Vosseberg, "The W-MODEL – Strengthening the Bond Between Development and Test," Proc. STAREAST 2002, 2002.
[13] S. Mathur and S. Malik, "Advancements in the V-Model," International Journal of Computer Application, vol.1, no.12, pp.29–34, 2010.
[14] J.O. Clark, "System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective," 3rd Annual IEEE International Systems Conference, pp.381–387, 2009.
[15] I. Londesbrough, "A Test Process for all Lifecycles," 2008 IEEE Conference on Software Testing Verification and Validation Workshop, pp.327–331, 2008.
[16] S. Gupta and M. Hussain, "A New Process Model for Embedded Systems Control for Telecom Industry," International Journal on Computer Science and Engineering, vol.4, no.9, Sept. 2012.
[17] B. Boehm, "Value-Based Software Engineering," Software Engineering Notes, vol.28, no.2, p.4, March 2003.
[18] R. Ramler, S. Biffl, and P. Grünbacher, "Value-Based Management of Software Testing," in Value-Based Software Engineering, ed. S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Grünbacher, pp.225–244, Springer, 2006.
[19] L. Huang and B. Boehm, "How Much Software Quality Investment Is Enough: A Value-Based Approach," IEEE Software, vol.23, no.5, pp.88–95, Sept./Oct. 2006.
[20] Q. Li, Y. Yang, M. Li, Q. Wang, B.W. Boehm, and C. Hu, "Improving software testing process: feature prioritization to make winners of success-critical stakeholders," Journal of Software: Evolution and Process, vol.24, no.7, pp.783–801, 2012.
[21] A. Cockburn, Agile Software Development: The Cooperative Game, 2nd ed., Addison-Wesley, 2006.
[22] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principlesof user-centered agile software development: A literature review," Information and Software Technology, vol.61, pp.163–181, 2015.
[23] C. Ardito, P. Buono, D. Caivano, M.F. Costabile, and R. Lanzilotti, "Investigating and promoting UX practice in industry: An experimental study," Int. J. Human-Computer Studies, vol.72, no.6, pp.542–551, 2014.
[24] M. Hassenzahl and N. Tractinsky, "User Experience – A Research Agenda," Behaviour and Information Technology, vol.25, no.2, pp.91–97, March-April 2006.
[25] L. Plonka, H. Sharp, P. Gregory, and K. Taylor, "UX design in agile: a DSDM case study," Agile Processes in software Engineering and Extreme Programming, Springer, LNBIP, vol.179, pp.1–15, 2014.
[26] F. Lizano, M.M. Sandoval, and J. Stage, "Integrating usability evaluations into scrum: a case study based on remote synchronous user testing," Human-Computer Interaction, Springer, vol.8510, pp.500–509, 2014.
[27] M. Sliger and S. Broderick, The Software Project Manager's Bridge to Agility, Addison-Wesley, 2008.
[28] J. Cheung, J. Scanlan, J. Wong, J. Forrester, H. Eres, P. Collopy, P. Hollingsworth, S. Wiseall, and S. Briceno, "Application of Value-Driven Design to Commercial AeroEngine Systems," Journal of Aircraft, vol.49, no.3, pp.688–702, 2012.
[29] ISO/IEC 25010:2011(E), "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," ISO/IEC 2011.
[30] K.C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering," IEEE Software, vol.19, no.4, pp.58–65, July/Aug. 2002.
[31] W. Edwards and F.H. Barron, "SMARTS and SMARTER: Improved Simple Methods for Multiattribute Utility Measurement," Organizational Behavior and Human Decision Processes, vol.60, no.3, pp.306–325, 1994.
[32] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.
[33] A. Ishizaka, "Clusters and pivots for evaluating a large number of alternatives in AHP," Pesquisa Operacional, vol.32, no.1, pp.87–101, 2012.
[34] A. Abran, J.W. Moore, P. Bourque, and R. Dupuis, SWEBOK – Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004.
[35] I. Sommerville, Software Engineering, 9th ed., Addison Wesley, 2011.
[36] R.S. Pressman and B.R. Maxim, Software Engineering: A Practitioner's Approach, 8th ed., McGraw-Hill, 2014.
[37] N. Kano, N. Seraku, F. Takahashi, and S. Tsuji, "Attractive Quality and Must-be Quality," Hinshitsu, The Journal of the Japanese Society for Quality Control, pp.39–48, April 1984.
[38] S. Faulk, D. Harmon, and D. Raffo, "Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering," Proc. First International Conference on Software Product Line Engineering, Aug. 2000.
[39] S. Amland, "Risk Based Testing and Metrics," 5th International Conference EuroSTAR '99, vol.53, no.3, pp.287–295, Nov. 1999.
[40] M. Felderer and R. Ramler, "Integrating Risk-Based Testing in Industrial Test Processes," Software Quality Journal, vol.22, no.3, pp.543–575, 2014.
[41] C. Catal and D. Mishra, "Test Case Prioritization: A Systematic Mapping Study," Software Quality Journal, vol.21, no.3, pp.445–478, 2013.
[42] M. Yoon, E. Lee, M. Song, and B. Choi, "A Test Case Prioritization through Correlation of Requirement and Risk," Journal of Software Engineering and Application, vol.5, no.10, pp.823–835, 2012.
[43] H. Srikanth, C. Hettiarachchi, and H. Do, "Requirements Based Test Prioritization Using Risk Factors: An Industrial Study," Information and Software Technology, vol.69, pp.71–83, 2016.
[44] Q. Li and B. Boehm, "Improving Scenario Testing Process by

Adding Value-Based Prioritization: An Industrial Case Study," Proceedings of the 2013 International Conference on Software and System Process, pp.78–87, 2013.

Youngsub Han is currently a Ph.D. candidate in Computer Science and Engineering at Korea University in Seoul, Korea. He works for Defense Agency for Technology and Quality in Korea, as a senior researcher. His research interests are requirements engineering, software testing, and software maintenance.

Dong-hyun Lee received his Ph.D. in Computer Science and Engineering from Korea University. His primary research interests are software engineering on embedded system, value-based software engineering, ubiquitous computing. He received his BS in electrical engineering and MS in computer science from Korea University.

Byoungju Choi received her Ph.D. degree in Computer Science from Purdue University. At present, she is a professor in Department of Computer Science and Engineering at Ewha Womans University in Seoul, Korea. Her research interests are in software engineering with particular emphasis on software testing, embedded software testing.

Mike Hinchey received the B.Sc. from University of Limerick, Ireland, M.Sc. from University of Oxford, UK and Ph.D. from University of Cambridge, UK. Previously he was Director of the NASA Software Engineering Laboratory and has been Full Professor or Visiting Professor in USA, Japan, Sweden, UK and Germany. He is currently Director of Lero-the Irish Software Research Centre and Professor of Software Engineering at University of Limerick.

Hoh Peter In received his Ph.D. degree in Computer Science from the University of Southern California (USC). He was an Assistant Professor at Texas A&M University. At present, he is a professor in Department of Computer Science and Engineering at Korea University in Seoul, Korea. He is an editor of the EMSE and TIIS journals. His primary research interests are software engineering, social media platform and services, and software security management. He earned the most influential paper award for 10 years in ICRE 2006. He has published over 100 research papers.