

# Applying the Value/Petri Process to ERP Software Development in China

LiGuo Huang Barry Boehm  
Computer Science Department  
University of Southern California  
Los Angeles, CA 90089-0781, USA  
001 213-740-6470

{liguohua, boehm}@usc.edu

Hao Hu Jidong Ge Jian Lü  
State Key Lab of Novel Software Tech.  
Institute of Computer Software  
Nanjing University, 210093, China  
+86 25-83593283

{myou, gjd, lj}@ics.nju.edu.cn

Cheng Qian  
Neusoft Co., Ltd.  
Neusoft Park  
Shenyang 110179, China  
+86 24-83667714

qc@neusoft.com

## ABSTRACT

Commercial organizations increasingly need software processes sensitive to business value, quick to apply, and capable of early analysis for subprocess consistency and compatibility. This paper presents experience in applying a lightweight synthesis of a Value-Based Software Quality Achievement (VBSQA) process and an Object-Petri-Net-based process model (called VBSQA-OPN) to achieve a manager-satisfactory process for software quality achievement in an on-going ERP software project in China. The results confirmed that 1) the application of value-based approaches was inherently better than value-neutral approaches adopted by most ERP software projects; 2) the VBSQA-OPN model provided project managers with a synchronization and stabilization framework for process activities, success-critical stakeholders and their value propositions; 3) process visualization and simulation tools significantly increased management visibility and controllability for the success of software project.

## Categories and Subject Descriptors

D.2.9 [Management]: Software process models, Software quality assurance (SQA), Life cycle

## General Terms

Management, Economics, Verification, Performance.

## Keywords

Software Quality, Value, Cost, ROI, Object Petri Nets (OPN), Software Process Formalization, Software Process Simulation

## 1. INTRODUCTION

Enterprise Resource Planning (ERP) is a business management system that integrates all facets of the business process, including planning, manufacturing, sales, and marketing. The booming

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
ICSE'06, May 20–28, 2006, Shanghai, China.  
Copyright 2006 ACM 1-59593-085-X/06/0005...\$5.00.

economy in China has encouraged the development of ERP software to improve various business operations such as inventory control, order tracking, customer service, finance and human resources. Attracted by the potential huge profit, more and more software companies are jumping into this field, which leads to severe competition among ERP solution providers. They are expected to provide continuous value realization for their success-critical stakeholders to survive the competition. These trends drive the changes in the characteristics of ERP software development in China as follows:

- A variety of stakeholders with different value propositions are involved in the entire software development life cycle.
- Product lines are maintained as a basis for future upgrades.
- Three process patterns (*deadline-driven*, *product-driven* and *market-trend driven*) are selectively applied in their software development life cycle based on different business cases.
- Different software quality assessment criteria are set based on different business cases by various project success-critical stakeholders. Thus different software development activities are adopted in different process patterns.

ERP solution providers in China have established their own sets of software development process activities. Most of the activities are based upon value-neutral approaches and/or the Waterfall model, which are difficult to adapt to changing characteristics and often lead to project failures.

## 2. VALUE-BASED SOFTWARE QUALITY ACHIEVEMENT PROCESS FRAMEOWRK

Value-Based Software Quality Achievement (VBSQA) process framework [1] is generated from the WinWin Spiral Model [2] and coupled with the theory of value-based software engineering [3]. It provides a top-level guideline to generate the process instance for achieving the stakeholder WinWin-balanced software quality requirements based on risk-driven concurrency. Instead of using one-size-fits-all metrics to measure software quality achievement, the VBSQA process framework enables its users to elicit success-critical stakeholders' value propositions (i.e., prioritization, expected & desired values) with respect to quality (Q-) attributes. It also helps identify their value conflicts on Q-attributes through risk analysis, architecture/technology evaluation and milestone reviews. In addition, it helps resolve the conflicts by performing tradeoff analyses to engineer the stakeholder WinWin-balanced Q-attribute requirements. Furthermore, the framework guides us to use real earned value to

monitor and control the progress toward achieving the Q-attribute requirements through each milestone reviews.

The top-level steps and anchor point stakeholder commitment milestones (bolded) of the VBSQA process framework are listed in Table 1.

**Table 1. Top-level steps of VBSQA process framework**

1.	Identify top-level mission objectives and stages – including software quality (Q-) objectives
2.	Perform project cost/benefit analysis – Estimate project budget – Develop results chain to identify success-critical stakeholders and their top-level value propositions
3.	Stakeholders negotiate mutually satisfactory (Win-Win) quality (and other) goals and relevant mission scenarios.
4.	Concurrently engineer top-level Q-attribute and other requirements and solution tradeoff spaces.
5.	Identify top-level Q-risks, execute risk-mitigation spirals.
6.	Develop system top-level design and initial Feasibility Rationale Description (FRD).
7.	<b>Hold Life Cycle Objective (LCO) Review</b> – <b>Pass: go to 8. Fail: go to 4.</b>
8.	Concurrently engineer detailed Q-attribute and other requirements and solutions; resolve risks.
9.	Develop system detailed design and detailed Feasibility Rationale Description (FRD).
10.	<b>Hold Life Cycle Architecture (LCA) Review</b> – <b>Pass: go to 11. Fail: go to 8.</b>
11.	Construct, test, and deploy system – Use the mission scenarios and Q-attribute requirement levels as progress metrics and test cases – <b>Core Capability Demo (CCD)</b> – Monitor progress and change requests; perform corrective actions
12.	<b>Initial Operational Capability (IOC) Readiness Review</b>

### 3. VBSQA PROCESS EXPERIENCE ON A REAL-WORLD PROJECT CASE STUDY

The case study was originated from a real-world project in Neusoft, one of the biggest solution providers and software companies in China. It produces software to support ERP in the fields including telecommunication, electric power, enterprise e-business, social insurance, finance, education, tax, and mobile Internet. We name all of these software products as ERP software in this paper.

Based on the above VBSQA process framework we conducted the initial exercise with 2 project managers from Neusoft and 2 facilitators from Nanjing University. And an undergoing ERP software project in Neusoft, which is to upgrade a Documents and Images Management System (DIMS)<sup>1</sup> from version 6.0 to 7.0, was used as a case study. The current 6.0 version of the DIMS software developed by Neusoft had been used in several departments of Chinese government for three years. Some

departments were going to change their database platforms. At the same time, they might also add, remove or update certain attributes in the DIMS 6.0 database schema. In this case, DB administrators needed to export all the data from the old databases and import them to the upgraded ones. With the common requirements of the DB administrators from various departments, Neusoft decided to upgrade the DIMS from 6.0 to 7.0 by adding a new capability of data migration. The objective of this exercise was to tailor the VBSQA process to the ERP software development activities in China.

#### 3.1 The New Process Learning Curve

Two project managers from Neusoft were given two-week series of tutorials on the VBSQA process framework and the WinWin Spiral model. Then we started to evaluate the effectiveness of the tutorials. Two project managers were to utilize the VBSQA process framework as a guideline to generate a value-based process instance based on the current ERP software development activities in order to achieve the WinWin balanced DIMS quality requirements from various project success-critical stakeholders. They developed a process instance composed of 22 ERP software development activities. During the discussion after the exercise, we detected 6 misplaced activities due to the misinterpretation of the process steps in the VBSQA process framework. We also identified 4 missing activities which should have been included in the process instance due to the misunderstanding of value-based approach and WinWin Spiral model.

This exercise and previous experience suggested that project managers typically had very short attention spans and low tolerance for new “methods” since they were usually very busy. They all suggested that we provide an easy-to-use process framework with a short learning curve.

Furthermore, project managers tended to use the VBSQA process as a guideline. They expected it to be able to adapt to the changes in the ERP software development activities and workflows. Thus, customization of VBSQA process framework toward specific software development, such as ERP software development, would improve its application and provide considerable value for project managers and software companies.

#### 3.2 Maintaining the Flexibility of the Process

The VBSQA process framework covers all the phases and milestones in the entire software development life cycle of the WinWin Spiral model. It also includes various software development activities to incorporate the value-based consideration.

On the one hand, for most Chinese ERP solution providers, different software quality assessment criteria are set based on different business cases so that different activities may be selected to meet them. Three different process patterns (*deadline-driven, product-driven and market-trend driven*) are usually applied in the software development based on different business cases. Deadline-driven business case applies when rapidly accommodating a few minor product upgrading requirements from one or two departments within an organization. Product-driven business case applies when accommodating a request to upgrade to the next version due to the aggregation of some common upgrading requirements from various departments. In this case, the quality of the upgraded product is the process driver

<sup>1</sup> The DIMS project is anonymous for the sake of commercial confidentiality.

rather than meeting a schedule. Market-driven business case applies when the upgrade of the product is driven by the market trend or rivals' products, for instance, a change from the Client/Server architecture to the Web-based architecture. In this case, providing superior capabilities to capture greater market share as early as possible is the key process driver. To meet the different requirements of different business cases, a flexible process generation platform should be created to enable the trim and/or addition of the steps/activities based on the VBSQA process framework.

### 3.3 Identifying Flaws in a Process Instance

Maintaining the flexibility of the VBSQA process framework might introduce flaws during process instance creation. While a process step and/or a software development activity in the process instance could be included or excluded by project managers, the created process instance might contain flaws and/or risks. For instance, the dropped activities might cause the violation of the critical path activity dependencies so that the precondition(s) of a specific activity could not be satisfied prior to its execution.

### 3.4 Tradeoffs among Conflicting Quality (Q-) Attributes

Software quality is an integrative concept consisting of a number of attribute dimensions such as availability, security, performance, evolvability, schedule and cost. Thus, achieving software quality is a multi-attribute decision problem. Each project success-critical stakeholder can define his/her expected and desirable levels for each Q-attribute. The WinWin-balanced quality requirements are the reassessment and compromise of the Q-attributes among success-critical stakeholders.

As reported by the project managers, there usually existed at least one pair of conflicting Q-attributes in many ERP software projects. In the DIMS upgrade project, Performance and Evolvability were a pair of conflicting Q-attributes. There were several architecture options to select from. Direct copy between DBs favored Performance in terms of both throughput and storage space at the cost of Evolvability. XML-based architecture favored Evolvability by accommodating future changes of DB platforms and schema at the cost of Performance. Developers had to select one feasible architecture from several options in order to balance the stakeholders' conflicting Q-attribute requirements. If we could identify the conflicting Q-attributes as early as possible in the software development life cycle, we would be able to mitigate the risk of project failure by performing tradeoff analysis among conflicting Q-attributes. Our exercise showed that the stakeholder interaction activities (e.g., *External Prototype Evaluation*, *Architecture options external review*, *Selected architecture external review*) in the software development process were more effective in identifying the conflicting Q-attributes and that the stakeholder negotiation activities were more effective in performing tradeoff analysis.

### 3.5 The Importance of Determining Project Stakeholders' Perspectives and Interaction Point(s)

Our exercise showed that not all the stakeholders were required to have the same level (intensity) or type of involvement [6] in every

activity in the software development process. Thus planning the level of involvement of each stakeholder was critical.

In the mean time, the project success-critical stakeholders' interaction activities could either mitigate the software quality risks or drive the changes of stakeholders' value-propositions. And the costs (i.e., activity cost, potential rework cost) are also associated and/or resulted from such activities. Thus, planning the activities for stakeholders' interaction and negotiation at different phases of the software development life cycle might result in different Return-On-Investment (ROI). For instance, the ROI of the *External Prototype Evaluation* activity and *Architecture Options External Review* in LCO stage might be different than the counterparts in LCA stage. Therefore, determining the time of the stakeholders' interaction activities in a process instance was very important.

## 4. MODELING VBSQA PROCESS USING OBJECT PETRI NETS (OPN)

To tackle the problems encountered in our first attempt at the VBSQA process training and exercise, we built a VBSQA process simulation tool VBSQA Process Generator which could be used for ERP software development in China. Some related works on process simulation have been investigated in [4, 5]. The purposes of process simulation modeling are discussed in [4]. And a discrete-time process simulator to support software project managers in task scheduling is presented in [5].

The overall structure of VBSQA Process Generator is shown in Figure 1. It is composed of three components: VBSQA Process Creator, VBSQA Process Checker and VBSQA Process Simulator. The simulation results can be utilized as a feedback to adjust and improve the current VBSQA process instance. Their application is illustrated using DIMS upgrade case study in section 5. It aims to help industrial practitioners visualize the process and generate an appropriate and optimized VBSQA process instance based on a certain project business case.

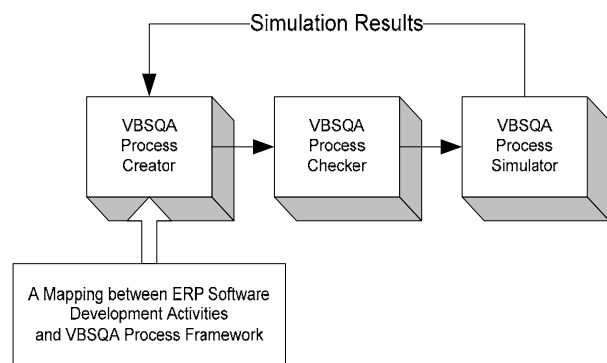


Figure 1. The overall structure of VBSQA Process

In order to build the process simulation tool, first we needed to model the VBSQA process using a process language that could capture its characteristics. Furthermore, the process language should be precise enough to eventually support verification and drive simulations which could help us address some problems identified in section 3.

## 4.1 Purpose of VBSQA-OPN Process Modeling

Value-based software development processes tend to be stakeholder-involved with a great deal of concurrency and backtracking. VBSQA process is one of these processes with the emphasis on achieving stakeholder WinWin-balanced software quality requirements. Thus, it usually involves multiple stakeholders with different value propositions on Q-attributes and different perspectives about the on-going process.

Object Petri Nets (OPN) [10], which is an extension of traditional Petri Nets, was chosen to model the VBSQA process. Based on the fact that the control structures of software processes are similar to those of programming languages, Osterweil proposed the idea of “Software processes are software too” [7]. Because the control structures of Petri Nets (PN) are similar in expression to programming languages, they can be used to model software processes [8]. Aalst has listed three reasons for using Petri Nets for process modeling and analysis: 1) formal semantics despite the graphical nature, 2) state-based instead of event-based, 3) abundance of analysis techniques [9]. Furthermore, PN has the merit of modeling concurrent process activities. As an extension of traditional PN, OPN inherits these merits of PN in process modeling.

In addition, OPN supports the separation of concerns among different stakeholders’ perspectives of the process by object oriented approach. Each stakeholder’s process instance can be modeled in a separate Object Net (ON) by inheriting the activities from the relevant process steps in the System Net (SN) (i.e., the VBSQA process framework). We took the “object-oriented” approach in the sense that the VBSQA process framework was modeled as a SN which was used as a process guideline. And each stakeholder’s process instance was modeled as an object that followed the workflow of the guideline to perform the ERP software development activities. Then the interaction and negotiation among stakeholders and the synchronization between each stakeholder’s ON and the SN could be defined later. Thus, OPN is able to adapt to the changes in the ERP software development activities and workflows.

VBSQA-OPN model provides a feasible solution to automation or semi-automation of the VBSQA process. Section 4.2 provides the formal definitions of our VBSQA-OPN process modeling.

## 4.2 Formal Definitions of VBSQA-OPN Process Modeling

In this section, we present the formal definitions of the VBSQA-OPN.

### Definition 1. Object Petri Nets (OPN)

A Petri net is a 3-tuple  $PN = (P, T, F)$ , where  $P$  is a finite set of places,  $T$  is a finite set of transitions,  $P \cap T = \emptyset$ ,  $F \subseteq (P \times T) \cup (T \times P)$  is a set of arcs, representing the flow relation between places and transitions. Tokens, representing the pre- and post- conditions of activating a specific transition, flow from one place to another. In the diagram, we use a circle to represent a place, use a bar to represent a transition and use a solid dot to represent a token. Please refer to [11] for the basic concepts of Petri nets.

An OPN is a 3-tuple  $OPN = (SN, ON_s, \rho)$ . This definition supports multi-objects and it is extended from the Valk’s definition [10].

- $SN = (P, T, W)$  is a Petri net, named System Net,  $W \subseteq (P \times T) \cup (T \times P)$ . The tokens in SN refer to the Object Nets defined below.
- $ON_s = \{ON_1, \dots, ON_n\}$  ( $n > 1$ ) is a finite set of Object Nets in OPN,  $ON_i = (B_i, E_i, F_i)$  is a Petri net, named Object Net.  $F_i \subseteq (B_i \times E_i) \cup (E_i \times B_i)$ .
- SN and  $ON_s$  synchronize via “channels” ( $\rho$ ).  $\rho$  is the synchronous relation between SN and  $ON_s$ ,  $\rho \subseteq T \times E$ , where  $E := \bigcup \{E_i \mid 1 \leq i \leq n\}$ .

To support multi-objects in VBSQA-OPN model, we extend a special Occurrence Rules based on the Valk’s Three Occurrence Rules [10], which are omitted here.

### Definition 2. VBSQA-OPN

VBSQA-OPN(SN, ON\_s,  $\rho$ ) is the modeling of VBSQA process based upon OPN, where

- $SN = (P, T, W)$  is the VBSQA process framework. Here, the transition set  $T$  represents the steps/milestones and it is divided into two disjoint subsets  $T_{syn}$  and  $T_{st}$ , where  $T = T_{syn} \cup T_{st}$ .
  - $T_{syn}$  is a set of synchronous transitions, which represent the process steps/milestones that are actually performed by stakeholder(s) in their process instances.
  - $T_{st}$  is a set of status transitions. A status transition can only immediately follow a synchronous transition in the SN.

The graphical representations of the two types of transitions are shown in the legend of Figure 2. The tokens in SN refer to Object Nets (i.e., stakeholders’ process instances) and point to the marking of Object Nets.

Here, SN can be either the entire VBSQA process framework  $SN_0$  or a tailoring from  $SN_0$  based on the project business cases. Given the set of transitions to be deleted, we can construct the tailored process framework SN as follows. The set of transitions to be deleted are denoted as DEL. When transition  $t$  is deleted,

- 1) Let  $DEL := \{t\}$ ,
- 2)  $DEL := DEL \cup \{\text{the status transitions immediately following } t\}$
- 3)  $\forall x \in DEL$ , if  $\exists i, x \in Loop_i$  then  $DEL = DEL \cup Loop_i$ .  $Loop_i$  denotes the set of transitions which belong to the same closed loop in the directed graph.
- 4) Delete the transitions in DEL and add the arc(s) from the places before DEL to the transitions following DEL.

- $ON_s = \{ON_1, \dots, ON_n\}$  ( $n > 1$ ) represents a set of process instances of stakeholders and  $ON_i = (B_i, E_i, F_i)$  is the process instance of stakeholder  $i$ . In  $ON_i$ , the transition set  $E_i$  represents the ERP software development activities that should be performed by stakeholder  $i$  and it includes three

disjoint subsets,  $E_{i_{auto}}, E_{i_{syn}}, E_{i_{st}}$ , where  $E_i = E_{i_{auto}} \cup E_{i_{syn}} \cup E_{i_{st}}$ .

- $E_{i_{auto}}$  is a set of the object-autonomous transitions, which represents the autonomous activities of stakeholder  $i$  that can not be mapped to any given step/milestone in  $SN$  ( i.e., VBSQA process framework ).
- $E_{i_{syn}}$  is a set of synchronous transitions, which can be mapped to the steps/milestones in  $SN$  ( i.e., VBSQA process framework ) and has synchronous relation with  $SN$ .
- $E_{i_{st}}$  is a set of status transitions, which can only immediately follow a synchronous transition.

The graphical representations of the three types of transitions are shown in the legend of Figure 3. The tokens in  $ON_i$  represent the Q-attributes (e.g., Performance, Evolvability, Schedule, Cost, etc.) concerned by the stakeholder  $i$ .

- $\rho = T_{syn} \times E_{syn}$  ( $E_{syn} := \cup\{E_{i_{syn}} | 1 \leq i \leq n\}$ ) defines the synchronous relation between  $SN$  and  $ON_s$ , that is, a mapping between the VBSQA process framework steps/milestones and ERP software development activities.
- Guard functions are defined to set the activation condition(s) for some transitions. In this case, a transition  $t \in T$  is activated in a marking  $M$  (denoted as  $M[t >]$ ) iff  $M \geq \bullet t$  and the  $t$  transition's guard functions are both satisfied.
- **Constraint 1.** The chronological order of ERP software development activities in the stakeholders' process instances  $ON_s$  is consistent with the chronological order of VBSQA process framework steps/milestones in the  $SN$  based on their mapping. For a  $VBSQA-OPN = (SN, ON_s, \rho)$ , if there is a path from step  $A_s$  to  $B_s$  in the  $SN$ , denoted as  $A_s \prec B_s$ , and in the  $ON_s$ , there exist two ERP activities  $A_o$  and  $B_o$  such that  $(A_s, A_o) \in \rho$  and  $(B_s, B_o) \in \rho$ , then there must exist a path from  $A_o$  to  $B_o$ , denoted as  $A_o \prec B_o$ .
- **Constraint 2.** Critical Path Activity Dependency. For a  $VBSQA-OPN = (SN, ON_s, \rho)$ , if transition  $A_s$  must be completed before transition  $B_s$  (i.e.  $A_s \prec B_s$ ) in the  $SN$ , (denoted as  $B_s \mapsto A_s$ ), and transition  $B_o$  exists in  $ON_i$ , that is,  $\exists B_o \in E_i, (B_s, B_o) \in \rho$  in  $ON_i = (B_i, E_i, F_i)$ , then  $\exists A_o \in E_j, (A_s, A_o) \in \rho$  in  $ON_j = (B_j, E_j, F_j)$  (denoted as  $B_o \mapsto A_o$ ).

## 5. APPLYING VBSQA PROCESS GENERATOR BUILT ON VBSQA-OPN MODEL

Based on the VBSQA-OPN Modeling of the VBSQA process, the VBSQA Process Generator was built. We asked two project managers to apply this tool on the DIMS upgrade case study. Section 5.1 shows how to use the VBSQA Process Creator to create the process instances for project success-critical stakeholders based on the VBSQA process framework. Section 5.2 illustrates how to identify the flaws of a process instance

based on defined process constrains in the VBSQA Process Checker. Section 5.3 presents some simulation results of the ERP VBSQA process.

### 5.1 VBSQA Process Creator: Creating an ERP VBSQA Process Instance

#### 5.1.1 Mapping the ERP Software Development Activities into VBSQA Process Framework

To shorten the VBSQA process learning curve and to reduce the flaws such as the misplacement of ERP development activities when creating a process instance, we mapped the ERP software development activities into each step/milestone in the VBSQA process framework. Table 2 shows a part of this mapping based on the current ERP software development activities. In the VBSQA-OPN model, VBSQA process framework was modeled as the System Net ( $SN$ ) and each stakeholder class's process instance was modeled as an Object Net ( $ON$ ) inherited from the  $SN$ . Note that we only distinguished different stakeholder classes in creating a process instance but not the various roles in one stakeholder class. For instance, we assumed that IV&V team and testing team belong to the Developers. Thus, to create a process instance for a stakeholder, project managers just needed to select a specific activity mapped into the VBSQA process step and added it into the plan for this stakeholder. The chronological orders of these activities were automatically inherited from the  $SN$ , which eliminated the process flaws of misplaced activities due to the misinterpretation of the process steps in the VBSQA process framework as discussed in section 3.1. And new activities which were not mapped into any step/milestone could be added into the stakeholders'  $ON$  as needed. Furthermore, if the ERP software development activities and/or workflows are changed in the future, we will only need to change the mapping.

#### 5.1.2 DIMS Upgrade Case Study: Creating the ERP VBSQA Process Instance

In the DIMS upgrade case study, we identified 4 stakeholder classes including *System Acquirer*, *DB Administrators*, *Software Maintainers* and *Developers*. Figure 2 shows a segment of the  $SN$  (i.e., VBSQA process framework). Figure 3 illustrates the corresponding segment of the  $ON$  representing a process instance for the Developers generated from the  $SN$ . Figure 4 illustrates the corresponding segment of the  $ON$  representing a process instance for the System Acquirer. Figure 5 shows an example of the creation of the Developer process instance using VBSQA Process Creator.

When the mouse cursor was rested over a particular process step/milestone of the  $SN$  in the VBSQA Process Creator as shown in Figure 5, the applicable procedure/approach, if any, was displayed in a textbox. In this way, a project manager could associate the procedure/approach to the specific activity mapped to this process step/milestone. Similarly, when the mouse cursor was rested over a particular activity in the  $ON$  representing the stakeholder process instance, the corresponding stakeholder responsibilities (e.g., the documents and/or product to be delivered) were displayed in a textbox. Therefore, by creating different process instances for various stakeholders, we could separate one stakeholder's responsibilities from others' with respect to the activities that he/she was involved in.

**Table 2. Mapping the ERP software development activities into VBSQA process framework steps/milestones**

VBSQA Process Framework Steps/Milestones (System Net)	ERP Software Development Activities (Object Nets)
Initiate project	Acquire system upgrade requirements (Developer)
Project cost/benefit analysis	Estimate system upgrade cost & develop DMR results chain (Developer)
	Verify system upgrade cost (System Acquirer)
SCS define acceptable & desired values for Q-attributes	Requirement elicitation meeting
	Groupware WinWin negotiation
Risk analysis & architecture/technology evaluation	Internal prototype evaluation (Developer)
	External prototype evaluation
Identify conflicting Q-attributes & perform tradeoff analysis	Identify conflicting Q-attributes & perform tradeoff analysis
SCS adjust acceptable values for Q-attributes	Stakeholder renegotiation
System top-level design and initial Feasibility Rationale Description (FRD)	System top-level design (Developer)
LCO Review	Architecture options internal review (Developer)
	Architecture options external review
SCS refine acceptable & desired values for Q-attributes	Requirement elicitation meeting
	Groupware WinWin negotiation
System detailed design and detailed Feasibility Rationale Description (FRD)	System detailed design (Developer)
LCA Review	Selected architecture internal review (Developer)
	Selected architecture external review
Core capability implementation	Core capability implementation (Developer)
Value-based core capability testing	Internal core capability testing (Developer)
CCD	Internal core capability demo (Developer)
	Onsite core capability demo
Remaining features implementation	Complete system implementation (Developer)
IOC Acceptance Review	Onsite System Acceptance Review

## 5.2 VBSQA Process Checker: Identifying the Flaws in a VBSQA Process Instance

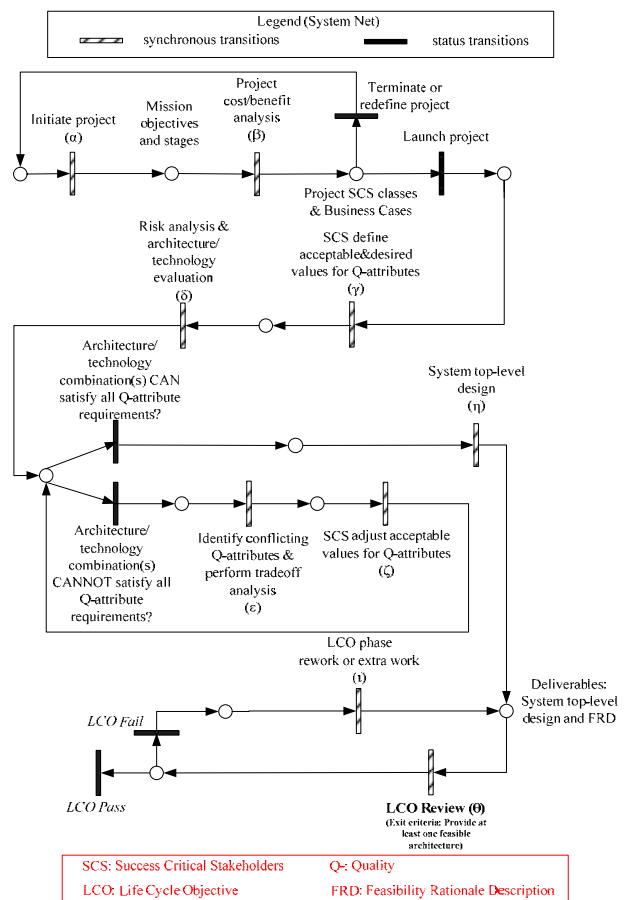
Based on the project business case, the project manager could choose to skip some steps in the VBSQA process framework during the ERP VBSQA process instance creation for success-critical stakeholders. That is, it allowed project managers to inherit a *NULL* activity from each step in the *SN* (i.e., VBSQA process framework). However, such flexibility provided by the tool could be both a strength and a weakness. It might introduce the flaws of missing activities which could cause the violation of critical path activity dependencies in a process instance.

One way to validate the process was to provide a process analysis capability to verify that critical path activity dependency constraints were not violated by the process definition. These

constraints were represented as formal properties defined in the VBSQA-OPN System Net (*SN*) and implemented in the VBSQA Process Checker. Some examples of the activity dependency constraints in the *SN* could be as follows:

- *SCS define acceptable & desired values for Q-attributes* must be completed before *Risk analysis & architecture/technology evaluation*
- *Risk analysis & architecture/technology evaluation* must be completed before *System top-level design*
- *System top-level design* must be completed before *LCO Review*
- *Value-based core capability testing* must be completed before *CCD*

And they needed to be translated into the precise formal definitions based on the Constraint 2 of Definition 2 in section 4.2.



**Figure 2. VBSQA-OPN System Net (SN): the LCO phase of VBSQA process framework**

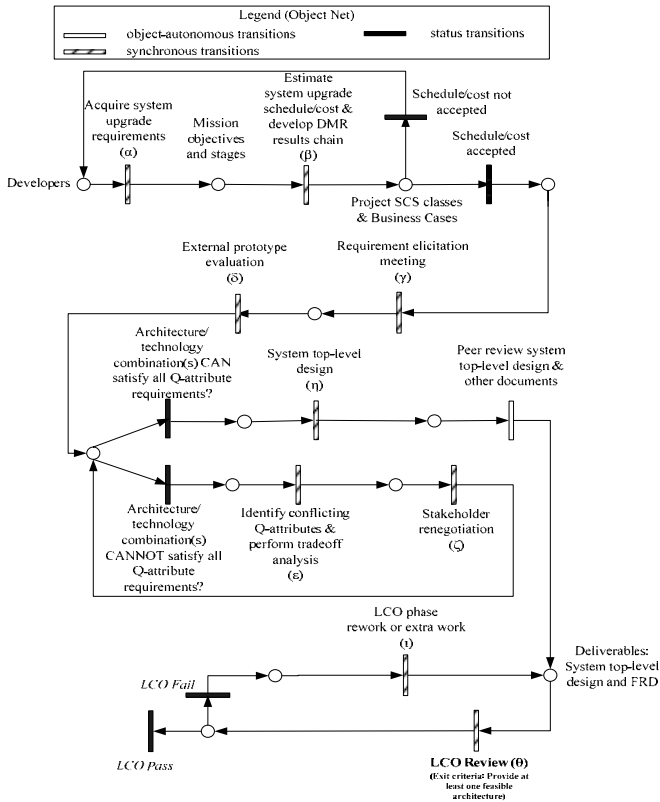


Figure 3. VBSQA-OPN Developer Object Net (ON): the LCO phase of the Developer process instance generated from the SN

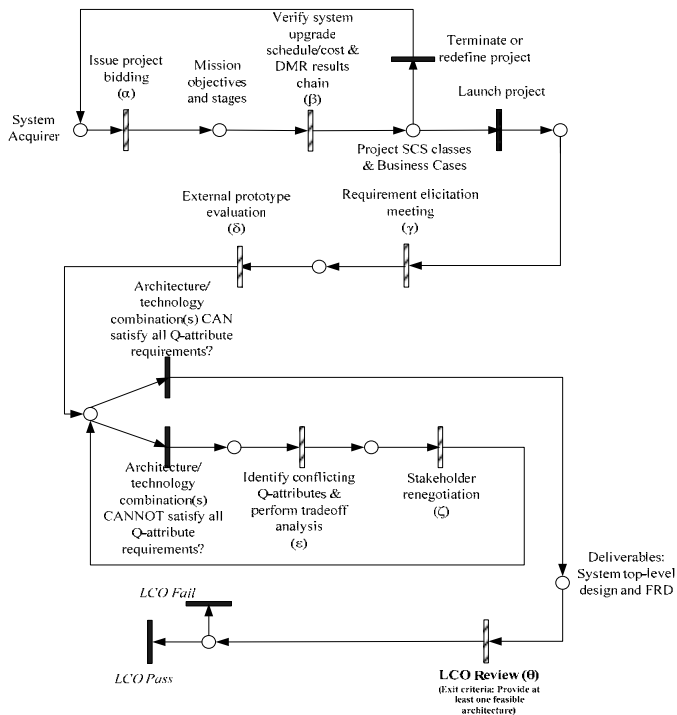


Figure 4. VBSQA-OPN System Acquirer Object Net (ON): the LCO phase of the System Acquirer process instance generated from the SN

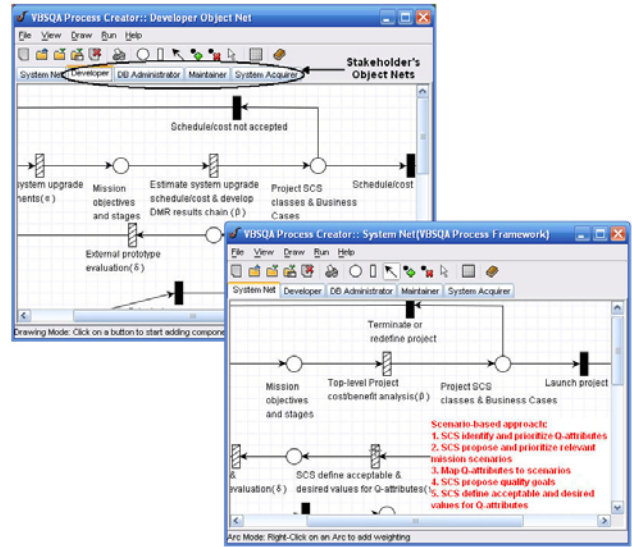


Figure 5. VBSQA Process Creator: VBSQA Process Framework (System Net) and the generated Developer process instance (Object Net)

For instance, the activity *System top-level design* had been planned in the LCO phase of the Developers' process instance. However, neither *Internal prototype evaluation* nor *External prototype evaluation* which were mapped to the *Risk analysis & architecture/technology evaluation* as shown in Table 2 was planned in any stakeholder's process instance. After analyzing the stakeholders' process instances in  $ON_s$  based on the defined critical path activity dependency constraints, the VBSQA Process Checker would display a warning message as "*Risk analysis & architecture/technology evaluation* must be completed before *System top-level design*".

### 5.3 VBSQA Process Simulator: Comparing the ROI of Synchronous Stakeholder Interaction Activities

The synchronous stakeholder interaction activities in the VBSQA process were usually effective in mitigating the software quality risks. At the same time, higher activity costs and different potential rework costs were incurred by such activities in different phases of software development. Thus, performing the stakeholder interaction activity at different phases of the software development life cycle might result in different Return-On-Investment (ROI). However, there lacked of quantitative analysis approaches and simulation tools to help project managers determine when to perform the synchronous stakeholder interaction activities during software development.

In this section, we present the simulation results of different ROI for various stakeholder interaction activities in different software development phases. The ROI was computed as following:

$$ROI = \frac{Value - Cost}{Cost} \quad (1)$$

### 5.3.1 Value Earned: the Synchronous Stakeholder Interaction Activity vs. Developer Internal Activity

Let's assume that totally  $m$  Q-attributes were identified for the project. And  $n$  ( $n=4$ ) software development phases were defined in the VBSQA process framework (LCO, LCA, CCD and IOC).

$$Value = \sum_{i=1}^m ((1 - \prod_{j=1}^n (1 - E_{ij})) \times R_i \times V) \quad (2)$$

$V$ : the total value of the project;

$E_{ij}$  (0-1): the effectiveness of a specific process activity on mitigating the risk of Q-attribute  $i$  if it is performed in phase  $j$ ;

$R_i$  (0-1): the risk of Q-attribute  $i$  to the total value of the project.

In DIMS upgrade case study, two project managers provided the inputs for each parameter based on their experience and expert judgement. Total value of the project was estimated as the contracted payment (\$50,000) that Neusoft would obtain from the system acquirer after the project was successfully completed by satisfying of all success-critical stakeholders' Q-attribute requirements. Totally 4 Q-attributes and their related operational scenarios were identified for this project as shown in Table 3.

**Table 3. DIMS upgrade project: Q-attributes and their risks to the project value ( $R_i$ )**

Q-attributes	Related Scenario	Priority	$R_i$ (0-1)
Performance	Complete data migration from the old DB to the upgraded DB within 1 day and within required storage space	High	0.8
Evolvability	Accommodate different DB platforms and schema in data migration	Medium	0.4
Schedule	—	High	—
Cost	—	High	—

We treated the failure of achieving a Q-attribute requirement as the risk to the total value of the project. Table 3 lists the risk of Performance and Evolvability to the project value, denoted as  $R_i$ .  $R_i$  is the product of the risk impact and the probability of risk occurrence. We defined the "risk impact" as *the proportion of the total project value that would have been lost if that Q-attribute risk had occurred*. Thus  $R_i$  was from 0 to 1. In this project, Performance was a High priority requirement from DB administrators and failure of achieving it would result in 80% loss of the project value. Evolvability was a Medium priority requirement from software maintainer and failure of achieving it would result in 40% loss of the project value. In this case study, we did not take into account the other two Q-attributes (Schedule, Cost) because they were not the major risks to this project.

Some steps in the VBSQA process framework such as Risk analysis & architecture/technology evaluation, LCO Review, LCA Review, CCD aimed to mitigate the quality risks. As shown in Table 2, two types of ERP software development activities were mapped to each of them, which were developer internal activities and synchronous stakeholder interaction activities. The examples of the former were *Internal prototype evaluation*, *Architecture options internal review*, *Selected architecture internal review*, *Internal core capability demo* as shown in the

right column of Table 2. These activities were accomplished only by developers without other stakeholders' participation. They were usually less effective in identifying Q-attribute risks. The examples of the latter were *External prototype evaluation*, *Architecture options external review*, *Selected architecture external review*, *Onsite core capability demo*. These activities were accomplished by developers and other success-critical stakeholder(s). Stakeholders were able to evaluate the prototype(s), review the architecture(s) or test the core capabilities together under the realistic operational environment. Thus, these activities were usually more effective in identifying Q-attribute risks. Furthermore, these activities at different phases of the software development life cycle also had different effectiveness in identifying Q-attribute risks. Table 4 shows the effectiveness of a specific process activity on mitigating the risk of Q-attribute  $i$  if it was performed in phase  $j$ , denoted as  $E_{ij}$ . Project managers provided the estimate of  $E_{ij}$  as the *proportion by which the risk of Q-attribute  $i$  would have been reduced if that process activity had been performed in phase  $j$* . Thus  $E_{ij}$  was from 0 to 1. Note that we treated the risk mitigation  $E_{ij}$  in its most general sense, which incorporated both the decrease of the probability of risk occurrence and their impact on the project value.

**Table 4. DIMS upgrade project: the effectiveness of developer internal activities vs. stakeholder interaction activities on Q-attribute risk mitigation ( $E_{ij}$ )**

Project Phases	Process Activities	Risk Mitigation ( $E_{ij}$ ) (0-1)	
		Performance	Evolvability
LCO (Life Cycle Objectives)	Architecture options internal review	0.2	0.2
	Architecture options external review	0.6	0.6
LCA (Life Cycle Architecture)	Selected architecture internal review	0.3	0.3
	Selected architecture external review	0.8	0.8
CCD (Core Capability Demo)	Internal core capability demo	0.2	0.2
	Onsite core capability demo	0.5	0.5
IOC (Initial Operational Capability)	Onsite system acceptance review	0.3	0.3

### 5.3.2 Cost: the Synchronous Stakeholder Interaction Activity vs. Developer Internal Activity

Two types of cost, the activity cost and the potential rework cost, were associated with a synchronous stakeholder interaction activity or a developer internal activity. The cost was computed as following:

$$Cost = \sum_{j=1}^n C_{aj} + C_r \quad (3)$$

$C_{aj}$ : the cost of a process activity at phase  $j$ ;

$C_r$ : the potential rework cost.

### 5.3.2.1 Activity Cost ( $C_{aj}$ )

The cost of the developer internal activity in DIMS upgrade project was estimated as \$500 by ERP software project managers. The synchronous stakeholder interaction activity usually had 2 or 3 time higher activity cost, estimated as \$1,500.

### 5.3.2.2 Rework Cost ( $C_r$ )

Whenever a Q-attribute risk was identified by a process activity, some amount of rework was needed as a remedy. Table 5 shows the potential rework cost  $C_r$  at 4 phases of VBSQA software development process. In the best case, rework was only needed for the current phase. However, sometimes rework extended to the previous phases. In the worst case, the rework needed to be done from the beginning of the project. The numbers in Table 5 show the rework cost  $C_r$  from phase(S) to phase(F). The numbers in the diagonal of Table 5 represent the rework cost within the LCO, LCA, CCD and IOC phases respectively. For instance, the cost of only reworking LCA phase was \$9,000, the cost of reworking LCA and CCD phases was \$35,000 and the cost of reworking LCA, CCD and IOC phases was \$46,000. Note that if developers needed to rework both the LCA and CCD phases because a risk was identified at the *Onsite Core Capability Demo*, the rework cost provided by project managers was \$35,000, which was larger than the sum of the rework cost within LCA and CCD phases (\$9,000 + \$19,000). They explained that since developers had to change the detailed architecture design and to redo the *Core Capability Implementation*, they usually needed extra effort to become familiar with the programming techniques for the new architecture design. Based on the ERP project managers, developer internal activities usually incurred little rework.

**Table 5. DIMS upgrade project: potential rework cost  $C_r$  at different phases of VBSQA software development process**

Rework Cost $C_r$ (\$)		Phase (F)			
		LCO	LCA	CCD	IOC
Phase (S)	LCO	3,000	12,000	38,000	50,000
	LCA	—	9,000	35,000	46,000
	CCD	—	—	19,000	31,000
	IOC	—	—	—	11,000

### 5.3.3 Simulation Results: ROI

Assuming that the synchronous stakeholder interaction activity (i.e., *Onsite System Acceptance Review*) was required in the IOC phase, we enumerated the possible combinations of stakeholder interaction activities and developer internal activities in the first three phases of software development life cycle (LCO, LCA and CCD). As shown in the second column of Table 6, LCO(i) denotes that we performed the developer internal activity (i.e., *Architecture Options Internal Review*) in the LCO phase. LCO(s) denotes that we performed synchronous stakeholder interaction activities (i.e., *Architecture Options External Review*) in the LCO phase. The same notation applies for other phases.

Given the inputs from two project managers, our simulation computed two ROI's for each process activity combination except the first one as shown in Table 6. One was for the worst-case scenario; the other was for the best-case scenario in terms of the potential rework cost. In the worst-case scenario, we assumed that rework happened after each synchronous stakeholder interaction

activity. And performing such activity in a certain phase could only avoid the future rework extending to this phase. For instance, only performing the *Selected Architecture External Review* in the LCA phase (i.e., LCO(i)\LCA(s)\CCD(i)\IOC(s)) would incur the rework on both LCO and LCA and another rework on both CCD and IOC. In the best-case scenario, we assumed that once we performed such activity in a certain phase the rework would only be needed from the beginning of the project to this phase and it could avoid all the future rework incurred by the Q-attribute risks afterwards.

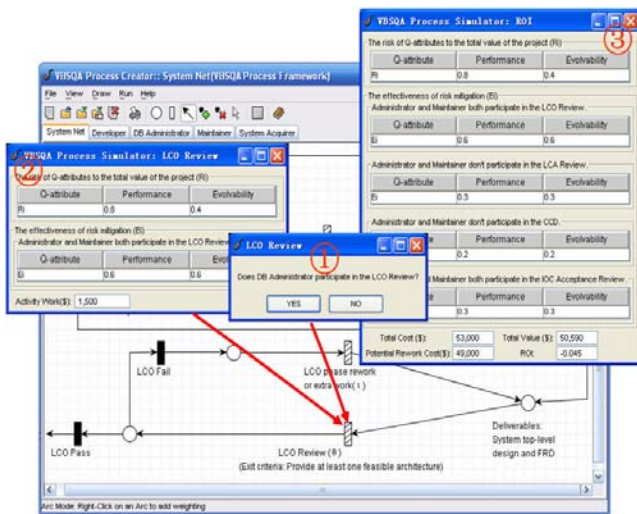
Even in the worse case scenario, *Selected Architecture External Review* in the LCA phase was particularly effective in improving the ROI since all the combinations with this activity (see 3, 5, 7, 8) in Table 6 produced relatively high worst-case ROI (0.162, 0.194, 0.205, 0.215). In the best-case scenario, *Architecture Options External Review* in the LCO phase was particularly effective in improving the ROI because we assumed that it avoided all the future rework incurred by Q-attribute risks after the LCO phase. Both results implied that performing synchronous stakeholder interaction activities in the architecting phase of a software project could produce higher return in terms of software quality risk mitigation. Figure 6 is a snapshot of using VBSQA Process Simulator to compute the ROI of a combination of stakeholder interaction activities and developer internal activities based on the System Net (i.e., VBSQA process framework).

**Table 6. Comparing the ROI of various combinations of synchronous stakeholder interaction activities and developer internal activities**

	Process Activity Combinations	ROI	
		Worst-Case Scenario	Best -Case Scenario
1	LCO(i) \ LCA(i) \ CCD(i) \ IOC(s)	-0.223	—
2	LCO(s) \ LCA(i) \ CCD(i) \ IOC(s)	-0.045	6.23
3	LCO(i) \ LCA(s) \ CCD(i) \ IOC(s)	0.162	2.414
4	LCO(i) \ LCA(i) \ CCD(s) \ IOC(s)	-0.09	0.149
5	LCO(s) \ LCA(s) \ CCD(i) \ IOC(s)	0.194	6.165
6	LCO(s) \ LCA(i) \ CCD(s) \ IOC(s)	0.002	5.765
7	LCO(i) \ LCA(s) \ CCD(s) \ IOC(s)	0.205	2.33
8	LCO(s) \ LCA(s) \ CCD(s) \ IOC(s)	0.215	5.48

### 5.3.4 Feedback on the ROI Simulation Results

The results in Table 6 enabled the project managers to rapidly assess the best-case and worst-case outcomes for their risk-mitigation decision options. Based on the feedback from two ERP project managers, the best-case scenario was usually not applicable in this project especially for the Performance attribute. In the LCO Review, developers usually could only provide the top-level system design and the non-functional prototype. Thus, the assumption that the *Architecture Options External Review* in the LCO phase could avoid all the future rework was too optimistic. However, in the LCA Review, when the detailed system design and the functional prototype were available, the assumption of avoiding the future rework after LCA phase was more applicable. Therefore, the real case scenario for Performance and Evolvability attributes in the DIMS upgrade project was closer to the worst-case scenario.



**Figure 6. VBSQA Process Simulator: Computing the ROI of a combination of stakeholder interaction activities and developer internal activities in an ERP VBSQA process**

As a result of this analysis, the DIMS project managers committed to hold a *Selected Architecture External Review* at the end of the LCA phase to evaluate the performance of selected XML architecture with totally 3,840,000 DB records. With the participation of DB administrators, software maintainers and developers in this activity, they identified the architectural risk on the Performance of data migration because the entire memory would be consumed by totally 97 intermediate XML files generated. After stakeholders' renegotiation, the developers re-architected the capability as Direct Copy with additional algorithms to only accommodate certain DB platforms and schema. Based on the project managers, without such analysis results they would have planned the process activities in a value-neutral way (e.g., holding *Selected Architecture Internal Review* at the end of the LCA phase only to save some activity cost) which would have increased the chance of project failure.

## 6. CONCLUSIONS AND LESSONS LEARNED

As we discovered in our application experiences of VBSQA process, solving a problem in theory and in practice were very different. In spite of the practical difficulties in applying a new process in software industry where traditional processes and methods dominated, the results showed that the application of value-based approaches was inherently better than the value-neutral ones that most ERP software projects employed in China.

In Microsoft Secrets [12], the ability to synchronize and stabilize multiple internal development teams is identified as a Microsoft critical success factor. The VBSQA-OPN model provided a framework in which the activities, value propositions, and commitments of multiple success-critical stakeholders could be synchronized and stabilized for a wide variety of process drivers.

The experience with the VBSQA Process Generator also told us process visualization and simulation tools significantly increased management visibility and controllability for the success of software project. In order to build such tools to visualize, verify and simulate the value-based processes involved by various stakeholders, the Object Petri Nets (OPN) provided a feasible solution to the value-based process modeling.

## 7. ACKNOWLEDGEMENT

This research is funded by the National Science Foundation in the US and by 863 Program (2004AA112090, 2005AA113160), 973 Program (2002CB312002) and NSFC (60233010, 60403014) in China. We would like to thank Weijie Zhu, Qiang Lin, Chao You and Fei Xiong in the Institute of Computer Software at Nanjing University, for their help in tool implementation. We would also like to express special thanks to DIMS project team in Neusoft Co., Ltd.

## 8. REFERENCES

- [1] L. Huang, "A Value-Based Process for Achieving Software Dependability", Proceedings of International Software Process Workshop, May, 2005, Beijing, China.
- [2] B. Boehm, W. Hansen, "Understanding the Spiral Model as a Tool for Evolutionary Acquisition", CrossTalk, May, 2001.
- [3] B. Boehm and A. Jain, "An Initial Theory of VBSE" in A. Aurum, S. Biffi, B. Boehm, H. Erdogmus, and P. Gruenbacher, Value-Based Software Engineering, Springer Verlag, 2005.
- [4] Kellner MI, Madachy RJ, Raffo DM: "Software process simulation modeling: Why? What? How?", Journal of Systems and Software, Vol. 46, No. 2/3, April, 1999.
- [5] F. Padberg, "A Software Process Scheduling Simulator," Proceedings of 25th International Conference of Software Engineering (ICSE'03), May, 2003,.
- [6] M. West, Real Process Improvement Using the CMMi, CRC Press, Feb. 1, 2004.
- [7] L. J. Osterweil, "Software Processes are Software too", Proceedings of International Conference of Software Engineering, 1987, pp. 2-13.
- [8] W. Deiters and V. Gruhn, "The FUNSOFT Net Approach to Software Process Management", International Journal on Software Engineering and Knowledge Engineering, 1994, pp. 229-256.
- [9] W. M. P. van der Aalst, "The Application of Petri Nets to Workflow Management", Journal of Circuits, Systems, and Computers, 1998, pp. 21-66.
- [10] R. Valk, "Petri nets as token objects: An introduction to elementary object nets", Proceedings of Application and Theory of Petri Nets, Springer-Verlag, 1998, pp. 1-25.
- [11] W. Reisig, Petri Nets, An Introduction, Springer Verlag, Berlin, 1985.
- [12] M. Cusumano and R. Selby, Microsoft Secrets, The Free Press, October, 1995.