

Modeling the Value-Based Software Process with Object-Petri-Nets*

Liguo Huang¹, Jidong Ge^{2,3}, Barry Boehm⁴, and Jian Lv²

¹ (Computer Science and Engineering Dept., Southern Methodist University,
Dallas, TX 75275-0122, USA)

² (State Key Lab of Novel Software Tech., Institute of Computer Software,
Nanjing University, Nanjing 210093, China)

³ (Software Institute, Nanjing University, Nanjing 210093, China)

⁴ (Center for Systems and Software Engineering, University of Southern California,
Los Angeles, CA 90089-0781, USA)

Abstract Commercial organizations increasingly need software processes sensitive to business value, quick to apply, supportive of multi-stakeholder collaboration, and capable of early analysis for subprocess consistency and compatibility. This paper and a companion paper “Applying Value-Based Software Process: An ERP Example” present our experience in applying a lightweight synthesis of a Value-Based Software Quality Achievement (VBSQA) process and an Object-Petri-Net (OPN) based process model to achieve a stakeholder win-win outcome for software quality achievement in an ERP software project in China. We attempt to answer such questions as (1) whether it is possible to model various project stakeholder perspectives using formal process modeling languages in a real-world project? (2) how to make stakeholders synchronize and stabilize their value propositions, activities and commitments as needed. The companion paper presents the VBSQA process and its application to the ERP project; this paper presents the OPN-based process modeling and its application. The application results of OPN process modeling confirmed that (1) the OPN-based process model provided project managers with a synchronization and stabilization framework for process activities, success-critical stakeholders and their value propositions; (2) process visualization and simulation tools significantly increased management visibility and controllability for the success of the software project.

Key words: software process modeling; object Petri nets (OPN); software quality; value-based software engineering (VBSE)

Huang LG, Ge JD, Boehm B, Lv J. Modeling the value-based software process with object-Petri-nets. *Int J Software Informatics*, 2010, 4(1): 101–119. <http://www.ijsi.org/1673-7288/4/i40.htm>

* This work is sponsored by the the National Science Foundation in the US and by the 863 Program (2004AA112090, 2005AA113160), 973 Program (2002CB312002) and NSFC (60233010, 60403014) in China.

Corresponding author: Liguo Huang, Email: lghuang@lyle.smu.edu

Received 2007-06-22; revised 2009-11-03; accepted 2010-04-19; published online 2010-04-28.

1 Introduction

One of the main reasons for the lack of correlation between an organization's level of investment in information technology (IT) and its success in the marketplace has been the decoupling of software and IT initiatives from other business initiatives critical to the generation of business value^[1]. This situation has been exacerbated by a generally value-neutral approach to software engineering, in which every requirement, use case, object, test case, and defect is considered to be equally important. The emerging field of value-based software engineering (VBSE)^[2] has arisen to provide frameworks and techniques to enable organizations to better incorporate value considerations into their software engineering practices. A successful application of the Value-Based Software Quality Achievement (VBSQA) process framework to a real-world ERP project has been presented in the companion paper^[16].

The Value-Based Software Quality Achievement (VBSQA) process framework^[15] is generated from the WinWin Spiral Model^[13,14] and coupled with the theory of value-based software engineering^[2]. It provides a top-level framework of steps for achieving a set of stakeholder WinWin-balanced software quality attribute requirement ranges, based on a risk-driven, concurrent-engineering approach. Instead of using one-size-fits-all, single-value parameters to define desired 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 in identifying their value conflicts on Q-attributes through risk analysis, architecture/technology evaluation and milestone reviews. In addition, it helps in resolving the conflicts by supporting tradeoff analyses to engineer the stakeholder WinWin-balanced Q-attribute requirements. Furthermore, the framework supports the use of real earned business value (in addition to the business value-neutral "earned value" systems in current use) to monitor and control progress toward achieving the Q-attribute requirements and associated business value realization. The detailed VBSQA process steps are elaborated in Ref.[16].

In general, value-based software processes explicitly indicate that different project stakeholders may have different concerns/priorities on process attributes and/or activities. To model value-based software processes it is desirable but often difficult to make all stakeholder classes with different value propositions use formal process modeling languages in a real software project. In addition, it would be inefficient and unreasonable if all stakeholders had been modeled in the identical way in fulfilling a common set of process activities/tasks. Stakeholders shall only synchronize as needed. With the above issues and challenges in mind, the following four research questions are proposed to guide our research.

RQ1 Is it possible to model various project stakeholder perspectives using formal process modeling languages in a real project?

RQ2 Can we separate concerns of project stakeholders in value-based software process modeling?

RQ3 While separating stakeholders' concerns, can we make them synchronize and stabilize their value propositions, activities and commitments as needed?

RQ4 Is it cost effective to use a formal process modeling language to model value-based software processes?

In this paper, we propose an Object-Petri-Nets (OPN) based approach to modeling

value-based software processes such as VBSQA. The OPN supports process modeling at two levels: the System Net (SN) and multiple Object Nets (ON). A value-based process framework (e.g. VBSQA process framework) can be modeled as the System Net and each stakeholder's process instance can be modeled in a separate Object Net (ON) by inheriting the activities from the relevant process steps defined in the System Net (SN). The negotiation and synchronization among stakeholders can be modeled via the interactions between the SN and ONs. In Ref.[3], we summarized the successful application of the VBSE process framework to the real-world Enterprise Resource Planning (ERP) project in China detailed in the companion paper. To facilitate its application, we modeled the VBSQA process framework and the stakeholders' process instances using Object Petri Nets (OPNs). Section 2 provides a description of the particular ERP project used to pilot the application of the particular VBSQA-OPN approach. Section 3 summarizes the OPN process framework used in the pilot project. Section 4 discusses the particular Object Petri Net (OPN) approach to model and implement the process, and the cost-benefit analyses of the OPN modeling in the pilot project. Section 5 discusses the main lessons learned from the OPN modeling, Section 6 compares the OPN with other process modeling languages in their capabilities of process modeling and discusses the threats to validity, and Section 7 concludes.

2 The Neusoft ERP-Domain Project and Value Propositions

2.1 VBSQA ERP pilot project nature and value propositions

The VBSQA Enterprise Resource Planning (ERP) pilot project involved a real-world project in Neusoft, one of the biggest ERP solution providers and software companies in China. Neusoft produces both infrastructure and applications software to support ERP in fields including telecommunication, electric power, enterprise e-business, social insurance, finance, education, tax, and mobile Internet. We classify all of these software products as ERP software in this paper.

The pilot project to apply the VBSQA process framework to a Neusoft ERP software development project was in the ERP infrastructure class. It involved the next release of a Neusoft ERP infrastructure capability to upgrade a Documents and Images Management System (DIMS) from version 6.0 to 7.0. The current 6.0 version of the DIMS software developed by Neusoft had been used in several departments of the Chinese government for three years. Some departments were going to change their database platforms. At the same time, they wished to 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 government departments, Neusoft decided to upgrade the DIMS from 6.0 to 7.0 by adding a new capability of data migration. The pilot project involved two project managers from Neusoft and two facilitators from Nanjing University in tailoring the USA-developed VBSQA process to this particular ERP infrastructure software development activity. Its value propositions were representative of such ERP infrastructure projects, as they placed high values on evolvability, performance, reusability, and reliability; and were less concerned with cost and schedule, as they could generally defer lower-priority features in order to meet next-release cost and schedule targets.

2.2 Challenges in getting started with the VBSQA process framework

Two project managers from Neusoft were given a two-week series of tutorials on the VBSQA process framework, the OPN System Net representation, and the WinWin Spiral model^[13,14], culminating with an activity to utilize the VBSQA process framework as a guideline to generate a value-based System Net 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 and evaluation on the effectiveness of the framework and tutorials based on 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. The following issues were discussed after the training session and exercise.

- Project managers needed process simplifications and checklists since they were usually very busy. They provided several suggestions for an easier-to-use process framework with a shorter learning curve.
- Automated process verification and validation (V&V) features were desirable to maintain the integrity of the generated process instances.

These were all incorporated into the OPN approach to representing and simulating the VBSQA System Net and Object Net representations, as discussed next.

3 Modeling VBSQA Process Using Object Petri Nets (OPN)

To address the four research questions proposed in Section 1 and tackle the problems encountered in our first attempt during the VBSQA process training and exercise, we built a VBSQA process simulation tool, the VBSQA Process Generator, which could be used for ERP software development. Some related perspectives on process simulation have been provided in Refs.[4, 5]. The purposes of process simulation modeling are discussed in Ref.[4], and a discrete-time process simulator to support software project managers in task scheduling is presented in Ref.[5].

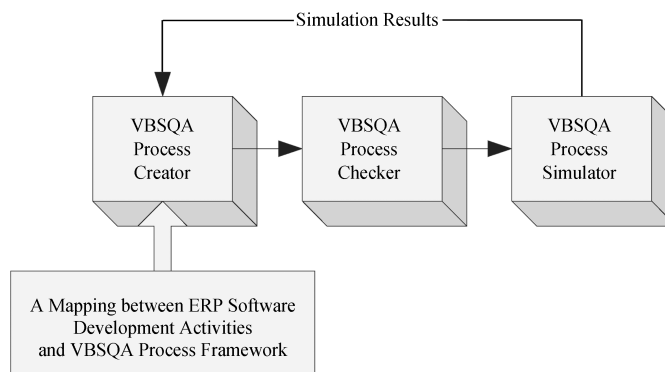


Figure 1. The overall structure of VBSQA process generator

The overall structure of the VBSQA Process Generator is shown in Fig.1. It is composed of three components: the 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 the DIMS Upgrade case study in Section 2. It aims to help industrial practitioners in visualizing the process and generate an appropriate and optimized VBSQA process instance based on a certain project business case.

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 illustrated in the companion paper. Furthermore, the process language should be precise enough to eventually support verification and drive simulations which could help us in addressing some problems identified in Section 2.2.

3.1 Purpose of VBSQA-OPN process modeling

Value-based software development processes tend to be stakeholder-involved with a great deal of concurrency and backtracking. The 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)^[9], 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”^[6]. Because the control structures of Petri Nets (PN) are similar in expression to programming languages, they can be used to model software processes^[7]. 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^[8]. Furthermore, PN had the merits of modeling concurrent process activities and of being familiar to the Neusoft personnel. 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 the 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. As shown in Fig.2, the interaction set between the system net and the object net defines the interaction relations between the VBSQA Process Framework and the Stakeholder Process Instances.

VBSQA-OPN model provides a feasible solution to automation or semi-automation of the VBSQA process. Section 3.2 provides the formal definitions of our VBSQA-

OPN process modeling.

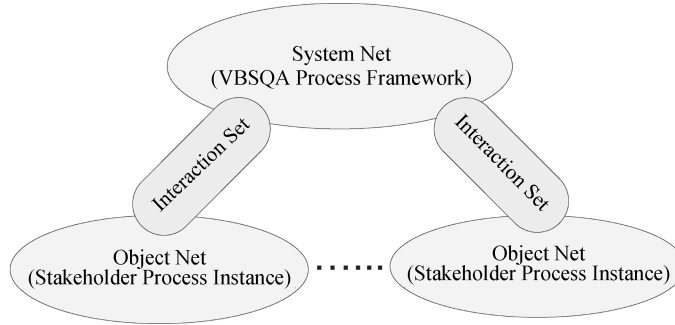


Figure 2. Modeling the VBSQA process on object Petri nets

3.2 Formal definitions of VBSQA-OPN process modeling

In this section, we present the formal definitions of the VBSQA-OPN. This paper applies the Valk's "Object Petri Nets"^[9] to model multi-view software process model. From the structure perspective, the object Petri net model includes two levels: system net and object net. There are interaction relation set between system net and object net. The most important idea in the model of object Petri nets is "Petri nets as token objects", that is, viewing the tokens of system net as the references of object nets.

Definition 1. Object Petri Nets (OPN) A Petri net is a 3-tuple, where P is a finite set of places, T is a finite set of transitions, $P \cap T = \phi$, $F \subseteq (P \times T) \cup (T \times P)$ 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 Ref.[10] for the basic concepts of Petri nets. An OPN is a 3-tuple. This definition supports multi-objects and it is extended from Valk's definition^[9].

An OPN is a 3-tuple. This definition supports multi-objects and it is extended from Valk's definition^[9].

- $SN = (P, T, W)$ is a Petri net, named System Net, $W \subseteq (P \times T) \cup (T \times P)$.
- $ON_s = \{ON_1, \dots, ON_n\} (n > 1)$ is a finite set of Object Nets in OPN , $ON_i = (B_i, E_i, F_i)$ is a Petri net, named Object Net. $F_i \subseteq (B_i \times E_i) \cup (E_i \times B_i)$. ON_i is an object referred by a token in SN .
- SN and ON_s synchronize via "channels" (ρ).
 - ρ is the interaction set between SN and ON_s , $\rho \subseteq (T \times E)$, where $E := \cup\{E_i | 1 \leq i \leq n\}$;
 - ρ_i is the interaction set between SN and ON_i , $\rho_i \subseteq (T \times E_i)$;
 - $\rho = \cup\{\rho_i | 1 \leq i \leq n\}$.

To support multiple object nets in VBSQA-OPN model, we define a set of Occurrence Rules extended from Valk's Three Occurrence Rules^[9] as follows:

Definition 2. Occurrence Rules for OPN A marking of an OPN is denoted (M, m_1, \dots, m_n) where M is a marking of the System Net SN and m_i is a marking of the Object Net ON_i . We define the three occurrence rules for OPN as follows:

- **System-autonomous firing:** A transition $t \in T$ is activated in a marking (M, m_1, \dots, m_n) of OPN if $\rho(t) = \phi$ and t is activated in M . Then the successor marking (M', m'_1, \dots, m'_n) is defined by $M \xrightarrow{t} M'$ (w.r.t. SN) and $(m'_1, \dots, m'_n) = (m_1, \dots, m_n)$, written as $(M, m_1, \dots, m_n) \xrightarrow{[t, \lambda]} (M', m'_1, \dots, m'_n)$ in this case. $\rho(t) = \phi$ means there is no transition in ON_s , synchronous with t . Here λ means empty action in ON_s .
- **Object-autonomous firing:** A transition e_{ij} [g1] ($\forall i, 1 \leq i \leq n, e_{ij} \in E$) [g2] is activated in a marking $(M, m_1, \dots, m'_i, \dots, m_n)$ of OPN if $\rho(e_{ij}) = \phi$ [g3] and e_{ij} [g4] is activated in m_i . Then the successor marking $(M', m'_1, \dots, m'_i, \dots, m'_n)$ is defined by $m_i \xrightarrow{e_{ij}} m'_i$ [g5] (w.r.t. ON_i) and $M' = M$ [g6], $\forall k \neq i, 1 \leq k \leq n, m'_k = m_k$ [g7] written as $(M, m_1, \dots, m_i, \dots, m_n) \xrightarrow{[\lambda, e_{ij}]} [g8] (M', m'_1, \dots, m'_i, \dots, m'_n)$ in this case. $\rho(e_{ij}) = \phi$ [g9] means there is no transition in SN synchronous with e_{ij} [g10]. Here λ means empty action in SN .
- **Synchronous firing:** a pair of transitions $\forall i \exists j, 1 \leq i \leq n, [t, e_{ij}] \in T \times E_i$ is activated in a marking (M, m_1, \dots, m_n) of OPN. If $\forall i \exists j, 1 \leq i \leq n, [t, e_{ij}] \in \rho$ and t and e_{ij} are activated in M and m_i , respectively. Then the successor marking (M', m'_1, \dots, m'_n) is defined by $M \xrightarrow{t} M'$ (w.r.t. SN) and $\forall i \exists j, 1 \leq i \leq n, m_i \xrightarrow{e_{ij}} m'_i$ [g11] (w.r.t. ON_i), written as $if [t, e_{ij}] \in \rho$ then $x_i = e_{ij}$ else $x_i = \lambda, (M, m_1, \dots, m_n) \xrightarrow{[t, x_1, \dots, x_n]} (M', m'_1, \dots, m'_n)$.

Definition 3. 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 composed of the top-level process steps and milestones. 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 Fig.3. The tokens in 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. $ON_s = \{ON_1, \dots, ON_n\} (n > 1)$ 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, $E_{iauto}, E_{isyn}, E_{ist}$, where $E_i = E_{iauto} \cup E_{isyn} \cup E_{ist}$.

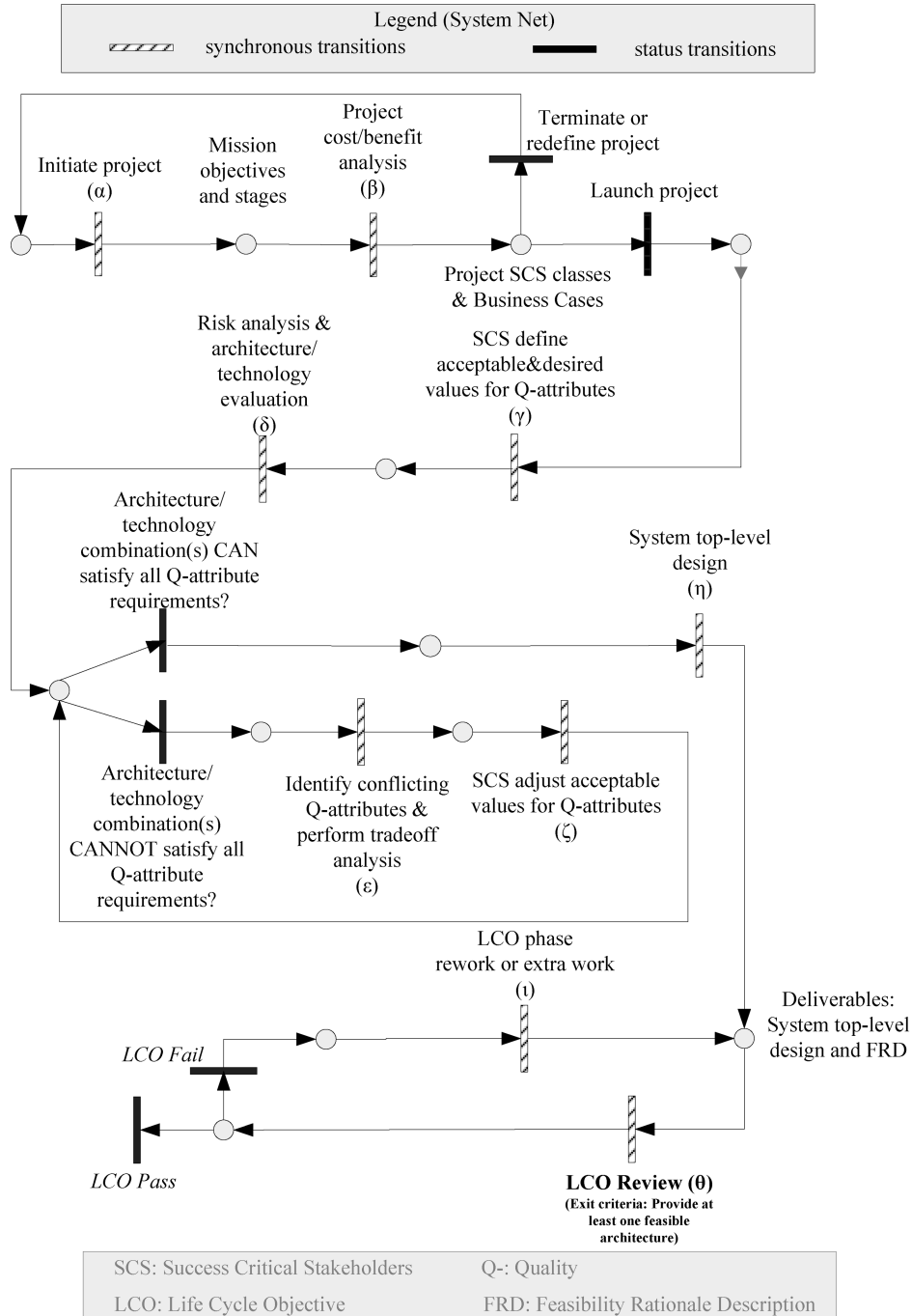


Figure 3. VBSQA-OPN system net (SN): the LCO phase of the VBSQA process framework

- E_{iauto} is a set of the **object-autonomous transitions**, which represents the autonomous activities of stakeholder i that cannot be mapped to any given step/milestone in SN (i.e., VBSQA process framework).
 - E_{isyn} 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_{ist} 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 Fig. 4. The tokens in ON_i represent the Q-attributes (e.g., Performance, Evolvability, Schedule, Cost, etc.) concerned by the stakeholder i .
- $\rho = T_{syn} \times E_{syn} (E_{syn} := \cup\{E_{isyn} | 1 \leq i \leq n\})$ 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 $M(p) \geq W(p, t), \forall p \in \bullet t$ [g12] 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 $B_s \vdash A_s$), 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 $B_o \vdash A_o$).

4 Applying the VBSQA Process Generator Built on the VBSQA-OPN Model

Based on the VBSQA-OPN modeling of the VBSQA process, the VBSQA Process Generator was built. We asked the two project managers to apply this tool on the DIMS Upgrade case study. Table 1 provides a summary how the general VBSQA process framework was instantiated for the DIMS Upgrade ERP project, which is detailed in the companion paper. Section 4.1 shows how the ERP VBSQA process instance was created using the OPN capabilities. Section 4.2 illustrates how the OPN capabilities help to identify the flaws of a process instance based on the defined process constraints in the VBSQA Process Checker. Section 4.3 presents the cost-benefit analysis results of VBSQA-OPN modeling on the DIMS Upgrade case study.

Table 1. Instantiating the VBSQA process framework steps for the DIMS upgrade project

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

4.1 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. 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.

To shorten the VBSQA process learning curve and to eliminate the flaws such as the misplacement of ERP development activities when creating a process instance, the ERP software development activities were mapped into each step/milestone in the

VBSQA process framework as shown in Table 1. 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. In order to create a process instance for a stakeholder, project managers just needed to select a specific activity mapped into the VBSQA process step and to add it into the plan for the stakeholder. Thus the chronological orders of these activities were automatically enforced by 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 2.2. In addition, new activities which were not mapped into any step/milestone could also 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 in Table 1. Fig.3 shows a segment of the SN (i.e., VBSQA process framework). Fig.4 illustrates the corresponding segment of the ON representing a process instance for the Developers generated from the SN. Fig.5 illustrates the corresponding segment of the ON representing a process instance for the System Acquirer. Note that the process instances of Developers and System Acquirer do not share the same set of activities. Fig.6 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 Fig.6, 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.

4.2 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 creating the ERP VBSQA process instance for success-critical stakeholders. That is, it allowed project managers to inherit a NULL activity from a certain 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 as described in Section 2.2, 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;

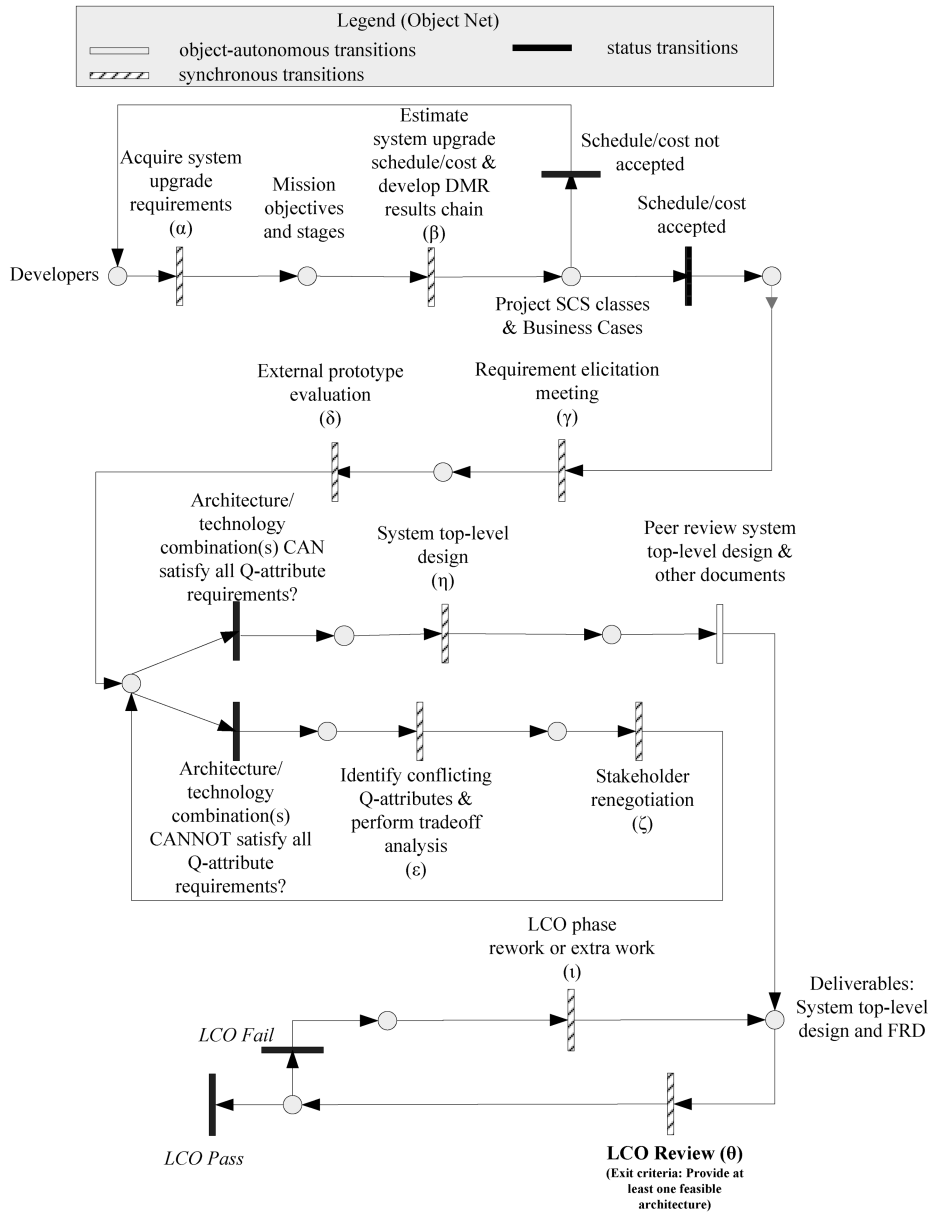


Figure 4. VBSQA-OPN developer object net (ON): the LCO phase of the developer process instance generated from the SN

And they needed to be translated into the precise formal definitions based on the

Constraint 2 of Definition 3 in Section 3.2.

For instance, the activity System top-level design has been planned in the LCO phase of the Developers' process instance. However, neither Internal prototype evaluation nor External prototype evaluation which are mapped to the Risk analysis & architecture-technology evaluation as shown in Table 1 is planned in any stakeholder's process instance. After analyzing the stakeholders' process instances in ONs based on the defined critical path activity dependency constraints, the ERP VBSDA Process Generator will display a warning message as "Risk analysis & architecture-technology evaluation must be executed before System top-level design". A segment of the Java code enforcing these activity dependency constraints in the ERP VBSDA Process Generator is shown in Fig.7.

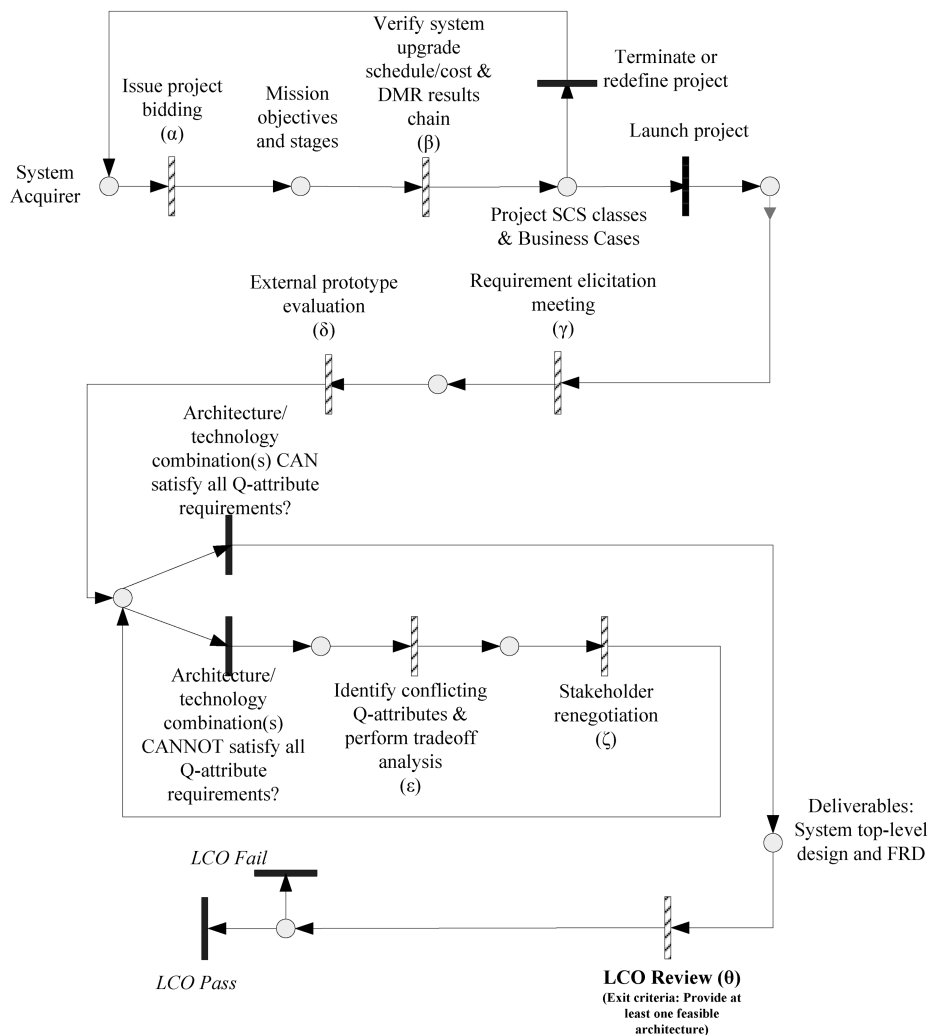


Figure 5. VBSQA-OPN system acquirer object net (ON): the LCO phase of the System Acquirer process instance generated from the SN

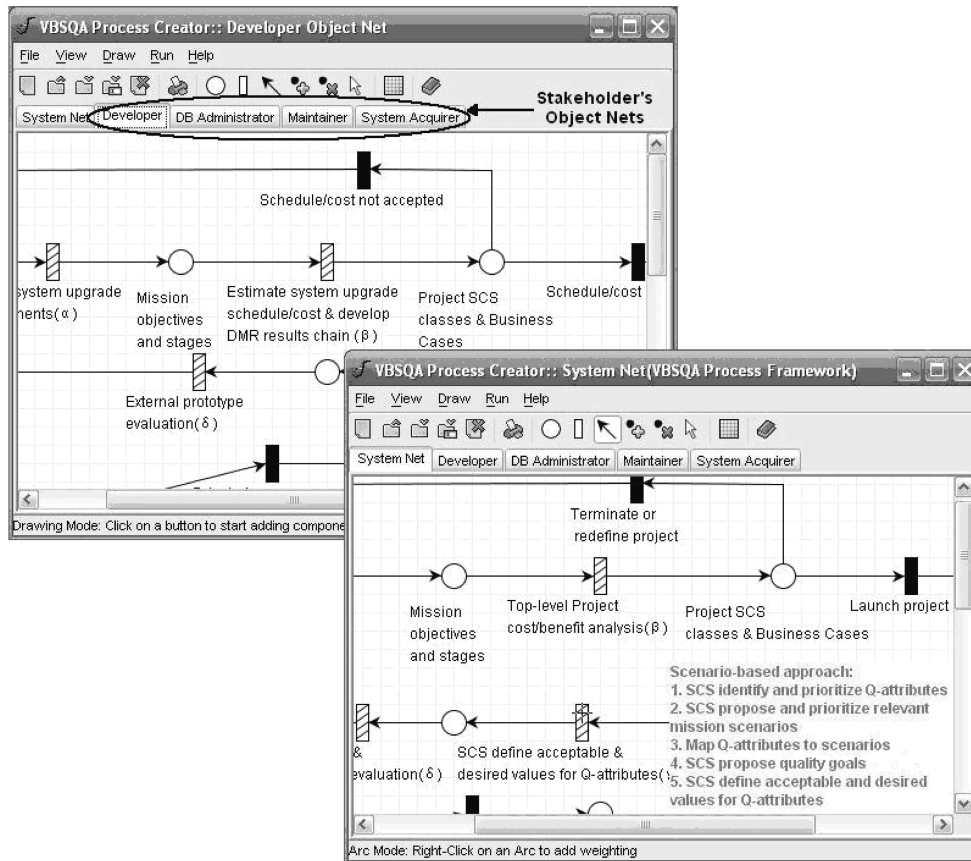


Figure 6. VBSQA process creator: VBSQA process framework (System Net) and the generated developer process instance (Object Net)

4.3 Cost benefit analysis of VBSQA-OPN modeling

The VBSQA-OPN modeling and application costs and benefits are analyzed based on the Neusoft DIMS Upgrade project, which are summarized in Fig.8. We have performed an initial cost benefit evaluation of the VBSQA-OPN modeling approach with two project managers involved in Neusoft DIMS Upgrade project. Neither of them had experience in value-based software processes or formal process modeling languages. The benefits are measured by the saved efforts (in hours) in terms of: 1) mutual learning; 2) developing project plan; 3) verification and validation (V&V) of project plan; 4) improving plan; 5) early vs. late plan rework We took the average of the two subjects in measuring the saved efforts.

The VBSQA-OPN Process Creator eased the development of project plans by mapping the ERP software development activities to the top-level VBSQA process steps and milestones. By adopting the VBSQA-OPN approach the project manager in average saved 56 hours in learning the WinWin Spiral Model and value-based process. 6 hours were saved in developing the process plan for DIMS upgrade project. Process instances generated by the VBSQA-OPN Process Creator eliminated the process flaws

of misplaced activities as indicated in Section 2.2. The flaws of missing activities were automatically identified by the VBSQA-OPN Process Checker. Thus another 6 hours were saved in verifying and validating the generated project plan. Although 3 hours were invested in optimizing the project plan based on the analysis results of the VBSQA-OPN Process Simulator detailed in the companion paper^[16], project managers indicated that they saved hundreds of hours in potential rework cost due to finding and avoiding sub-optimized arrangements of stakeholder interactive activities in the project plan.

```

//if other activities directly or indirectly depend on select
if (select.getDependencyOn().size() != 0)
{   String sMessage = select.getName();
    sMessage += " must be completed before ";
    for (int i = 0; i < select.getDependencyOn().size(); i++)
    {   Transition t = (Transition) select.getDependencyOn().get(i);
        sMessage += t.getName();
        sMessage += " ";
    }
    sMessage += ".";
    //Add warning message
    sMessage += "\nIf you want to delete anyway, " + "it will also delete the activities that\n"
        + "directly or indirectly depend on this activity.\n";
    sMessage += "Continue anyway?";
    int result = JOptionPane.showConfirmDialog(null, sMessage, "warning",
        JOptionPane.YES_NO_OPTION);
    //Delete select and all the activities that directly or indirectly depend on it
    if (result == 0)
    {   super.actionPerformed(e);
    } else {
        return;
    }
}

```

Figure 7. Enforcing the critical path activity dependency

5 Summary of Modeling Experiences and Lessons Learned

5.1 Continuing evaluation and refinement

The initial usage of the VBSQA-OPN process support system resulted in several process description defects due to misunderstandings and misinterpretations of the system's capabilities. Our evaluation of these led to several user suggestions that project managers needed some process simplifications and checklists since they were usually very busy. Their suggestions for an easier-to-use process framework with a shorter learning curve were incorporated into the OPN approach to representing and simulating the VBSQA System Net and Object Net representations.

5.2 Maintain 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 value-based considerations.

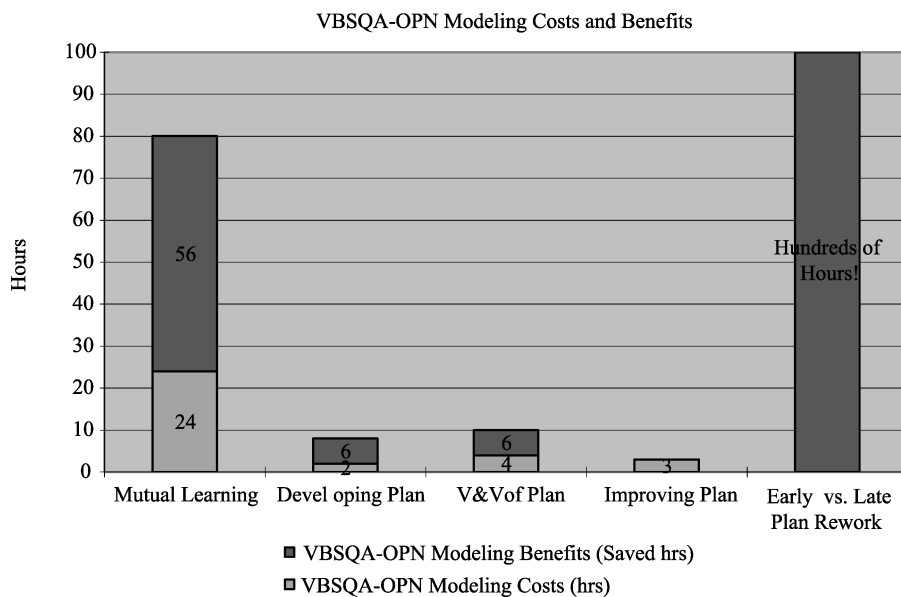


Figure 8. VBSQA-OPN modeling costs and benefits

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 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.

Driven by various business cases, project managers expected the VBSQA process to be able to adapt to the specifics and the changes in the ERP software development activities and workflows. Thus, customization of VBSQA process framework toward specific software development domains, such as ERP software development, would improve its application and provide considerable further value for project managers and software companies.

5.3 Need to identify flaws in a process instance

Maintaining the flexibility of the VBSQA process framework might introduce

flaws during process instance creation. When 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. Thus, although one would like a flexible modification capability, one would also like the formal consistency checking capabilities to remain, and to be able to operate incrementally and interactively to provide feedback on introduced flaws.

6 Discussion

Other process modeling languages such as Little Jil^[17] and System Dynamics^[18] may be used to model value-based software processes from different perspectives. For instance, Little Jil provides broad and precise semantics with growing analysis tool sets to support process verification and validation and resource allocation. System Dynamics as a Macro approach addresses high level value modeling and analysis^[19]. However, none of the existing research has been done on separating the concerns of different project stakeholders while enabling their interactions and synchronization on jointly performed process activities/tasks in value-based software process modeling.

In the case study, we have showed a successful application of the OPN-based approach on the VBSQA process generated from iterative WinWin Spiral process model. One threat to external validity is whether and how different types of process models can affect its application effectiveness and efficiency. We plan to reduce this threat by conducting further evaluation on other types of process models. The threats to internal validity include the representativeness of subjects selected for the case study.

7 Conclusions

Variety of stakeholders with different value propositions are usually involved in the entire software development life cycle. In applying the VBSQA process framework, different software quality assessment criteria are set based on business case priorities of various project success-critical stakeholders. Stakeholders have different levels of concerns on quality attributes so that they focus on software different development activities. With the VBSQA-OPN modeling approach and a case study of its real-world application, we can now address the questions that we proposed to guide this research.

RQ1 *Is it possible to model various project stakeholder perspectives using formal process modeling languages in a real project?*

RQ2 *Can we separate concerns of project stakeholders in value-based software process modeling?* The OPN process modeling approach provides a flexible platform to generate the VBSQA process instances by separating the concerns of individual stakeholder on the VBSQA process activities. The case study shows how VBSQA Process Creator models the VBSQA process framework as the System Net (SN) and each stakeholder class's process instance as an Object Net (ON) inherited from the SN in Section 4.1. This also addresses the 1st and 2nd issues discussed in Section 2.2 in that adopting the VBSQA-OPN modeling approach shortens the VBSQA process

learning curve and eliminates the flaws such as the misplacement of ERP development activities when creating a stakeholder process instance via the mapping between ERP project activities and VBSQA process framework as shown in Table 1.

RQ3 *While separating stakeholders' concerns, can we make them synchronize and stabilize their value propositions, activities and commitments as needed?* VBSQA-OPN modeling approach supports the coordination among various stakeholders' process instances through synchronous stakeholder interaction activities between the SN and ONs. 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.

RQ4 *Is it cost effective to use a formal process modeling language to model value-based software processes?* We have performed an initial cost benefit evaluation of the VBSQA-OPN modeling approach with two project managers involved in Neusoft DIMS Upgrade project without any experience in value-based software processes or formal process modeling languages. The benefits are measured by the saved efforts (in hours) in terms of: 1) mutual learning; 2) developing project plan; 3) verification and validation (V&V) of project plan; 4) improving plan; 5) early vs. late plan rework. Furthermore, to address the 2nd issue discussed in Section 2.2, the chronological order and critical path activity dependency constraints defined in VBSQA_OPN (see Section 3.2 Definition 3) and built into the VBSQA Process Checker enable us to automatically identify flaws in the generated VBSQA process instance for different stakeholders. In addition, the users of the VBSQA Process Generator also commented that the process visualization and simulation tools significantly increased their management visibility and controllability of the software project. We would also conclude that the OPN modeling approach is inherently compatible with value-based software development and maintenance processes. It enables different process views for various success-critical stakeholders while supporting the coordination among various stakeholders' process instances through synchronous stakeholder interaction activities.

Acknowledgment

We would like to express special thanks to the DIMS project team in Neusoft Co., Ltd., and to the IJSI reviewers of this paper for excellent improvement suggestions.

References

- [1] Thorp J. The Information Paradox. McGraw Hill, 1998.
- [2] Biffi S, Aurum A, Boehm B, et al. Value-Based Software Engineering. Springer Verlag, 2005.
- [3] Huang L, Boehm B, Hu H, et al. Applying the Value/Petri Process to ERP Software Development in China. Proc. of 28th International Conference of Software Engineering, 2006. 502–511.
- [4] Kellner M, Madachy R, Raffo D. Software process simulation modeling: Why? What? How? Journal of Systems and Software, 1999, 46: 91–105.
- [5] Padberg F. A Software Process Scheduling Simulator. Proc. of 25th International Conference of Software Engineering, 2003. 816–817.
- [6] Osterweil LJ. Software Processes are Software too. Proc. of 9th International Conference of Software Engineering, 1987, 2–13.
- [7] Deiters W, Gruhn V. The FUNSOFT Net Approach to Software Process Management. Inter-

- national Journal on Software Engineering and Knowledge Engineering, 1994, 4: 229–256.
- [8] van der Aalst WMP. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 1998, 8: 21–66.
 - [9] Valk R. Petri nets as token objects: An introduction to elementary object nets. *Proc. of 19th International Conference on Application and Theory of Petri Nets*, 1998. 1–25.
 - [10] Reisig W. *Petri Nets, An Introduction*. Springer Verlag, Berlin, 1985.
 - [11] Clements P, Kazman R, Klein M. *Evaluating Software Architectures*. Addison Wesley, 2002.
 - [12] Cusumano M, Selby R. *Microsoft Secrets*. The Free Press, October, 1995.
 - [13] Boehm B, Egyed A, Kwan J, et al. Using the WinWin Spiral Model: A Case Study. *IEEE Computer*, 1998, 31(7): 33–44.
 - [14] Boehm B, Hansen W. Understanding the Spiral Model as a Tool for Evolutionary Acquisition. *CrossTalk*, May, 2001.
 - [15] Huang L. A Value-Based Process for Achieving Software Dependability. *Proc. International Software Process Workshop*, 2005. 108–121.
 - [16] Huang L, Boehm B, Hu H, et al. Applying Value-Based Software Process: An ERP Example. *International Journal of Software and Informatics*, 2008, 2: 1–15.
 - [17] Cass A, Lerner B, McCall E, et al. Little-jil/juliette: a process definition language and interpreter. *Proc. of the 22nd International Conference on Software Engineering*, 2000. 754–757.
 - [18] Madachy R.J. *Software Process Dynamics*. IEEE Press and John Wiley & Sons, 2008.
 - [19] Osterweil L, Madachy R, Huang L, et al. Presentations on the Workshop of Modeling Systems and Software Engineering Processes, University of Southern California, 2008.