Assessing 3-D Integrated Software Development Processes: A New Benchmark*

Mingshu Li

State Key Lab of Computer Science and Lab for Internet Software Technologies,
Institute of Software at Chinese Academy of Sciences,
No. 4 South Fourth Street, Zhong Guan Cun, Beijing 100080, China
mingshu@iscas.ac.cn

Abstract. The increasing complexity and dynamic of software development have become the most critical challenges for large projects. As one of the new emerged methodologies to these problems, TRISO-Model uses an integrated three-dimensional structure to classify and organize the essential elements in software development. In order to simulate and evaluate the modeling ability of TRISO-Model, a new benchmark is created in this paper, called *SPW-2006 Example*, by extending the ISPW-6 Example. It may be used to evaluate other software process models, and/or to evaluate software organizations, software projects and also software development processes, particularly 3-D integrated software development processes. With the *SPW-2006 Example* and its evolution for quantitative evaluation to 3-D integrated software development processes, a new approach of *TRISO-Model based assessment and improvement* is enabled.

1 Introduction

Software Process Workshop (SPW) provides an annual forum for assessing current and emerging software process capabilities, and for obtaining insights into worthwhile directions in software process research. TRISO-Model (TRidimensional Integrated SOftware development Model), presenting a 3-D integrated software engineering methodology, was proposed in the SPW 2005 held in Beijing, China [1]. Its main objective is to deal with the problems caused by the increasing complexity and dynamic in current software development projects.

Process simulation is an effective mechanism for the study of the complexity and dynamic of software development processes and has attracted the focus of both research and industry communities. An expression of the attraction is the annual workshop on software process simulation modeling (ProSim) from 1998, a leading event for the simulation and modeling of software processes. In May 2006, ProSim will be held jointly with SPW in Shanghai, China, co-locating with ICSE 2006.

A software process benchmark is used to understand the current status of a software project, to evaluate its modeling or the current practice gaps to the benchmark, and to identify further process improvement opportunities. An assessment is used to examine a software organization's processes against a reference model to determine

* Supported by the National Natural Science Foundation of China (Grant Number: 60573082).

Q. Wang et al. (Eds.): SPW/ProSim 2006, LNCS 3966, pp. 15 – 38, 2006.

the processes' capability or the organization's maturity, and to meet its quality, cost, and schedule goals.

In order to evaluate and improve integrated software development processes, this paper puts forward a new process benchmark; and presents a new process assessment and improvement approach.

The remainder of the paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces TRISO-Model semantic specifications. Section 4 creates a new process benchmark *SPW-2006 Example* for effective evaluation of integrated software development processes. Section 5 presents a new approach of *TRISO-Model based assessment and improvement*. The paper is concluded in Section 6 with a summary and directions for future work.

2 Related Work

From the last 80s, some process modeling languages and corresponding tools, such as Little-JIL [2] and ADELE-TEMPO [3], have been designed to provide precise and comprehensive ways to represent various software process elements. Cost estimation methods, such as COCOMO II [4] and Web-COBRA [5], are invented to gain better predictability and quantitative control from the perspective of economics. Boehm's recent work on Value-Based Software Engineering [6] tries to further integrate value considerations into all of the existing and emerging software engineering principles and practices. The Personal Software Process (PSP) [7] and People Capability Maturity Model (P-CMM) [8] stress the factor of people, and provide a guide towards developing, motivating, and organizing the work force.

As Reifer lists in [9], the top challenges for nowadays developments fall into a large variety of interwoven areas such as technology, people, economy, change management and so on. CMMI [10] provides a framework covering most factors related to software development. MBASE [11] proposes a framework for avoiding model clash among different models (i.e. Process Model, Product Model, Property Model, and Success Model) in software development.

In SPW2005, the latest achievements on integrating different aspects of software development, besides TRISO-Model, are also presented by some researchers. Estublier relates processes, software production and humans in a pyramid framework to show and contrast the new and original potential uses of process technology [12]. Rombach proposes integrated software process & product lines (SPPL) [13] as a systematic way to choose both artifacts and processes needed for a given project. Osterweil [14] and Warboys [15] suggest different angle of views to integrate microprocess and macroprocess, respectively.

A simulation model is a computational model that represents an abstraction or a simplified representation of a complex dynamic system [16]. It offers the possibility of experimenting with different management decisions. Kellner et al. cluster the many reasons for using processes simulations into six categories of purpose, including [17]: strategic management, planning, control and operational management, process improvement and technology adoption, understanding, and training and learning.

Continuous modeling and discrete modeling are the two main approaches to build models in the simulation domain [18]. A continuous simulation model represents the

interactions between key process factors, as a set of differential equations, where time is increased step by step. On the other hand, discrete modeling is based on the metaphor of a queuing network, where time advances when a discrete event occurs. Continuous modeling and discrete modeling only enhance the analysis of some aspects of the process, at a cost to other aspects [19]. A software process, however, shows both discrete system aspects (start/end of an activity, reception/release of an artifact by an activity) and continuous system ones (recourse consumption by an activity, percentage of developed product, percentage of discovered defects). It would be desirable to use a continuous modeling for the dynamic environment, and a discrete one for tasks and resources [20]. A combined model would allow investigation of the effects of discrete resource changes on continuously varying productivity. There are some other simulation techniques, like state based process models, rule based languages, petri nets [21], and agent-based simulation [22].

A benchmark is a test or set of tests used to compare the performance of alternative tools or techniques [23]. It usually has three components: motivating comparison, task sample and performance measures. A proto-benchmark is a set of tests that is missing one of these components. The most common proto-benchmarks lack a performance measure and are sometimes called case studies or exemplars. These are typically used to demonstrate the features and capabilities of a new tool or techniques. A software process benchmark is an average reference value that the process statistically performs in a given sector or a given region [24].

For the purpose of making comparisons between different software process technologies, the ISPW-6 Example [25] was proposed as a benchmark problem at the 6th International Software Process Workshop. It has been used successfully to exam the essential features of some main software process methods in the last 90s, e.g., OPSIS [26] applies a view mechanism to graph-based process modeling languages of type Petri-net; MVP-L1 [27] is oriented towards process-modeling-in-the-large to concentrate on the formalization of interrelations between individual processes; MERLIN [28] uses a PROLOG-based language as a basis of the process definition. It was later extended to incorporate teamwork and process change (ISPW-7)[29].

The ISPW-6 Example is mainly designed for assessing the software process modeling approaches, and as a reasonable simplification, pays less attention to some other software development critical factors. However, “change” is a much more complex problem in real-world software development [30]. Because of the complexity, even though the problem caused by requirements changes has been noticed quite a long time ago, it is still one of the most frequent reasons for project failure. Nowadays the paradigm has shifted to be driven by a set of interwoven factors, such as technology, management, quality, knowledge, and economic considerations, so some extensions should be made to the ISPW-6 Example from process-oriented perspective to a multi-perspective framework. Relevant factors, such as economy, technology, and human, as well as the interactions among these factors should be incorporated into the framework.

SPEM (Software Process Engineering Metamodel) [31] is a software process modeling standard put forward by OMG (Object Management Group). In SPEM, a common syntax and structure for software development process [32] is provided based on the abstraction of process models such as RUP. As an extension of UML [33], SPEM inherits the expressiveness and popularity. With the graphic notations, SPEM offers a

comprehensive and documented view of the process model, which facilitates the communication of process stakeholders.

As a standard proposed by OMG aiming to be the unified software process modeling language, SPEM is being widely accepted. However, as a description language, the disciplines related to project management and analysis, process automation, etc. have not been involved. Furthermore, the dynamic semantics has not been addressed in SPEM.

MOF (Meta Object Facility) [34] is the meta-meta-model provided by OMG as the unified standard for domain metamodeling, and it provides common abstract syntax and semantic definition mechanism. MOF is suitable for constructing an integrated model of multi-dimension factors. However, a metamodel constructed in the MOF-based metamodeling method, as well as UML and SPEM, is an informal metamodel which has no precise semantics. Thus it is necessary to map it into another description using some formal method to reduce the ambiguity.

Figure 1 illustrates a segment (Review Design) of ISPW-6 Example represented by SPEM.

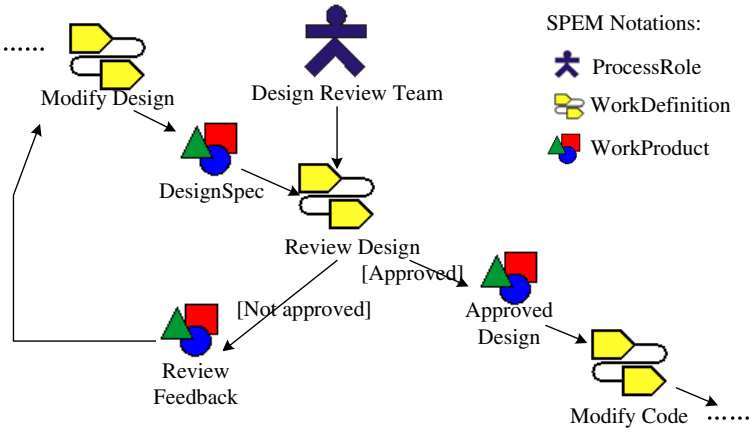

Fig. 1. Segment of ISPW-6 Example Represented by SPEM

CSP is developed by Hoare to address the concurrency and non-determinism in computing systems [35]. The basic idea is that those systems can be readily decomposed into subsystems which operate concurrently and interact with each other as well as with their common environment. As for the software process, Greenwood tentatively introduces CSP as a tool to model the software process [36]. LOTOS, another process algebra language similar to CSP, is employed to separate a whole software process into several concurrent subprocesses executed by different actors in [37]. But the actors and artifacts are just treated as communication channels, so it is difficult to present more information about those elements.

Using CSP, the segment (Review Design) of ISPW-6 Example in Figure 1 can be specified as follows:

$$
\begin{aligned}
&ISPW6Part = ModifyDesign \,||\, ReviewDesign\\
&ModifyDesign = assigntodesigner \rightarrow designspec \rightarrow modifydesign\\
&\qquad \rightarrow modifieddesign \rightarrow ModifyDesignLoop\\
&ModifyDesignLoop = assigntodesigner \rightarrow designspec \rightarrow reviewfeedback\\
&\qquad \rightarrow modifydesign \rightarrow modifieddesign \rightarrow ModifyDesignLoop\\
&ReviewDesign = modifieddesign \rightarrow assigntoreviewteam \rightarrow reviewdesign \rightarrow\\
&\qquad (approved \rightarrow approveddesign \rightarrow reportmeasurementdata \rightarrow SKIP \mid\\
&\qquad notapproved \rightarrow reviewfeedback \rightarrow ReviewDesign)
\end{aligned}
$$

3 TRISO-Model and Its Semantic Specifications

Based on the *Technology-Process-Human* triad conception and successful software engineering methodologies in the past, TRISO-Model presents a 3-D integrated methodology for software development processes, i.e. software development processes should be integrated improved from three perspectives of *technology*, *process*, and *human*. This expanded view incorporates the benefits gained from integrations among technologies, processes and humans.

TRISO-Model classifies the essential elements of the software development process into three dimensions: SE Technology, SE Process and SE Human. From the viewpoint of TRISO-Model, a software development process is thought of as a process driven by the interactions among the entities in the three dimensions. The entities may be abstracted to the activities for SE Process, the actors for SE Human, and the input/output artifacts for SE Technology respectively. The interactions are modeled in Figure 2 as six integrations: (1) Development Integration; (2) Process Integration; (3) Service Integration; (4) Data Integration; (5) Management Integration; and (6) Use Integration. The former three are internal integrations; and the latter three are external integrations.

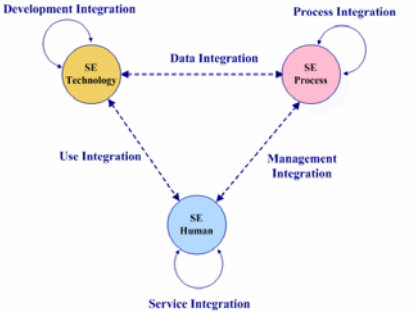

Fig. 2. Integrated Framework of TRISO-Model

3.1 Static Semantic Specification of TRISO-Model

The static semantics of TRISO-Model is represented by the elements of the entities in the three dimensions and the relationships among them. A static structure of TRISO-Model is shown as Figure 3.

Figure 4 illustrates the core concept of SPEM. The main idea is that a software development process is a set of collaborations among ProcessRoles that perform WorkDefinitions in which the WorkProducts are operated.

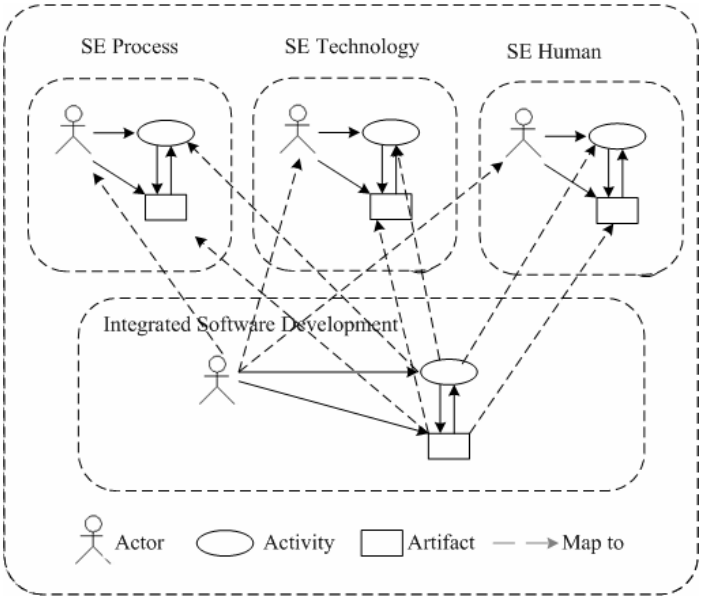

Fig. 3. Static Structure of TRISO-Model

The metaclasses in SPEM, ProcessRole, WorkDefinition and WorkProduct, may be viewed as the abstracted elements of *human* (or *actor*), *process* (or *activity*), and *technology* (or *artifact*) in TRISO-Model as shown in Figure 3. However, SPEM is over-simplified so that it cannot provide enough support to the integrated methodology. It has to be extended to describe the elements of entities in SE Human, SE Process and SE Technology dimensions of TRISO-Model and their relationships.

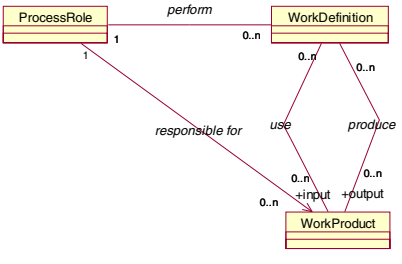

Fig. 4. Core Concept of SPEM

In TRISO-Model, an integrated soft-ware development process is expected to relate to the three dimensions. As an example shown in Figure 3, for each *actor* in the integrated process, there are corresponding *actors* in SE Process, SE Technology and SE Human dimensions; and it is the same with the *activities* and *artifacts*.

To support the idea stated above, we extend SPEM to *Integrated SPEM* (*I-SPEM* for short; more details will be introduced in another paper) with the metaclasses enhanced in three dimensions of SE Human, SE Process, and SE Technology. *I-SPEM* is defined as a M2 layer metamodel based on MOF, in which integrated elements in three dimensions and their relationships are specified in a consistent method as illustrated in Figure 5.

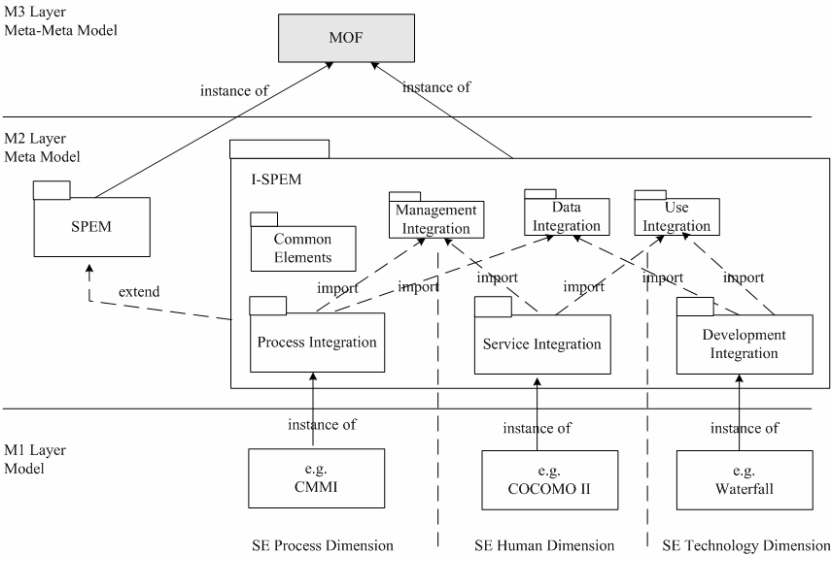

Fig. 5. Illustration of Static Semantics of TRISO-Model

In *I-SPEM*, there are six integration packages defined with the same names corresponding to Figure 2. The concepts and their relationships in three dimensions are defined in three internal integration packages; the integrations among the dimensions are defined in three external integration packages. The common elements and facilities are defined in the *Common Elements* package.

MOF, the M3 layer meta-metamodel, provides a consistent semantic base for every dimension specific metamodels in the framework of TRISO-Model. M1 layer models, the lowest abstraction level, are the instances of *I-SPEM* and the combination of discrete-type simulation, continuous-type simulation and analytical model. In Figure 5, CMMI, COCOMO II and Waterfall model are chosen only as the examples for the three dimensions. A different organization may have other choices. For instance, the ISO 9001 may be chosen to replace the CMMI.

3.2 Dynamic Semantic Specification of TRISO-Model

The dynamic semantics of TRISO-Model is represented by the evolutions of the entities in the three dimensions, and the communications and/or the coordination among them. A dynamic structure of TRISO-Model is shown as Figure 6. The *activities*, *actors*, and *artifacts* are the essential entities of the corresponding SE Process, SE Human, and SE Technology dimensions. Each entity has its own pattern of behaviors. It may communicate with other entities in the same dimension and/or coordinate with other entities in different dimensions through the synchronizations on some specific events.

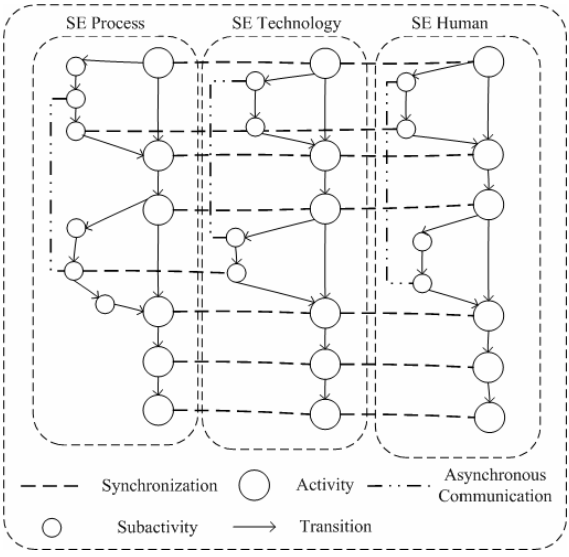

Fig. 6. Dynamic Structure of TRISO-Model

In the dynamic semantics, all the entities of TRISO-Model are mapped to three types of basic CSP processes, which are the *activity*, *actor*, and *artifact* processes. The three processes representing the three dimensions are combined together and coordinated by the synchronizations on some particular events. Each dimension is represented by a CSP process formed by the internal dimension integrations of the corresponding type of basic CSP processes. It can be taken as an agent possessing the necessary knowledge about how to evolve itself forward and having the exposed interfaces to synchronize with other dimensions.

Additionally, to make the content more comprehensible, all the illustrations presented in this section exclusively centralize on one problem, i.e., an abstracted software development process. The process is composed of the "requirement" and "design" activities only. The "requirement" activity begins with the requirements analysis and ends with outputting the requirements specification. In the "design" activity, the requirements specification is firstly input and then the system is designed. These two activities are sequentially arranged as those in the Waterfall lifecycle model. If the "requirement" activity and "design" activity are represented by two CSP processes, named $Requirement$ and $Design$ respectively, then the software development process can be denoted as:

$$DevelopemntProcess = req : Requirement; des : Design \tag{1}$$

To fully describe TRISO-Model, several aspects should be considered as the extensions on CSP, i.e. *CSP Extensions for TRSIO Model* (*CTM* for short; and will be discussed more in another paper). Firstly, CSP has to be extended to include asynchronous communication. In CSP, Hoare has chosen synchronized communications as basic. The synchronized communication means that a receiver blocks until a compatible agent is ready to send. Furthermore, CSP allows bi-party communications only.

Expression (1) cannot be implemented based on the synchronized communications, but it can be modeled by the following asynchronous communication. Secondly, in order to model the across dimension collaboration of TRISO-Model, a collaboration operator is needed. Two processes will be synchronized on the automatically inserted dummy actions indicated by the operator. Finally, a process in CSP is the behavior pattern of an entity. But the entity may have some attributes other than actions. In *CTM*, a process is extended with attributes that can be accessed by other processes.

The extension of asynchronous communication does not change the rule that the communications between processes are purely based on synchronizations. A buffer process is implicitly introduced between two communicating processes. The existence of the buffer process is transparent to the specifier. He or she can input or output any object freely without the need to considering the synchronization between the inputs and outputs. But the interactions among the inputting, outputting, and buffering processes are based on the synchronized communications. Here we recur to a *set* process to ease some constraints imposed by the buffer process. The *set* process, acting as the storage media, is one component of the environment. A *set* process based on the subordination operator is presented in [38].

Let the operator, !!, represent the asynchronous output. When using this operator, the specified object will be put into the set used for containing the artifacts of a software development process.

A *CTM* process outputting something to the environment can be set as:

$$\mathrm{Re}\,quirement_{CTM} = requirementanalysis \rightarrow !!requirementspec \rightarrow SKIP \tag{2}$$

where the subscript $_{CTM}$ means that the expression uses some notations that are defined in the *CTM*.

The above expression can be equivalently expressed in CSP notations as:

$$Requirement_{CSP} = (set : SET //$$
$$(requirementanalysis \rightarrow set.add!requirementspec \rightarrow SKIP))$$

where SET represents the *set* process and add is the channel used for inserting an object into the set.

Let the operator, ??, represent the asynchronous input. It means that the process needs an input from the buffered *set* process. As an example, the design activity may use the $requirementspec$ produced in (2):

$$Design_{CTM} = ??requirementspec!y \rightarrow design \rightarrow SKIP \tag{3}$$

where, $??requirement!y$ means retrieve $requirementspec$ from the *set* process and put the result into the variable y. (3) can also be described in CSP notations as:

$$Design_{CSP} = (set : SET //$$
$$(set.isin!requirementspec \rightarrow set.result?y \rightarrow design \rightarrow SKIP))$$

where $isin$ is a channel used for retrieving an object and the result can be got from the $result$ channel. A minor change may be made on the definition of the *set* process presented in [38]. A *NIL* or a referenced object should be returned from the $result$ channel instead of a Boolean value. When the needed object $requirementspec$ does not exist, the *set* process returns *NIL*. The inputting process will be blocked and retested later.

The three dimensions are represented by *CTM* processes in TRISO-Model. The software development process is the combination of the three processes. It is the synchronizations among the three processes that seamlessly integrate the three dimensions. In the dynamic semantics of TRISO-Model, the alphabets of the three dimensions are not obliged to have common actions. Thus each dimension can be separately defined in a divide-conquer strategy. The synchronization is carried out on the automatically inserted dummy actions. It is implemented by the following collaboration operator.

The operands of the collaboration operator are two *CTM* processes. The operator, modeling the external integrations of TRISO-Model, is denoted as:

$$\Theta_{synchronizationname\{\langle label1,label2\rangle,...\}}$$

where

synchronizationname the name of the synchronization
$\{\{label1,label2\},...\}$ a set of synchronized points
$\{label1,label2\}$ a tuple representing a point
$label1,label2$ the lable of the subprocesses that should be synchronized in the operands

This operator ensures that each step of the two processes is synchronized on the automatically inserted dummy actions.

As examples, two processes are defined as:

$$P = P1;P2;P3 \quad \text{,and} \quad Q = Q1;Q2;Q3$$

then,

$$P\Theta_{utilize\{\{p1,q1\},\{p3,q3\}\}}Q =$$
$$((p1.uitilize.q1 \rightarrow p1:P1);P2;(p3.uitilize.q3 \rightarrow p3:P3)) \parallel$$
$$((p1.uitilize.q1 \rightarrow q1:Q1);Q2;(p3.uitilize.q3 \rightarrow q3:Q3))$$

Here the representation of a dummy action is composed of the corresponding subprocesses, the name of synchronization, and the dots. But this does not violate the atomic property of an action in *CTM*.

It should be noted that two successive collaboration operators are syntactically legitimate. The synchronization points are the unions of the two operators. In this sense the two operators meet the communicative law.

4 3-D Integrated Software Engineering Process Benchmarking

As stated in Section 2, ISPW-6 Example is not competent in the face of the complex system nowadays, in which multi-dimension issues have to be considered. As the extension and improvement to ISPW-6 Example, we design a new benchmark, *a 3-D Integrated Software Engineering Modeling Example Problem*. It may be used to evaluate the emerging integrated software development models/methodologies like TRISO-Model. In correspondence with ISPW-6 Example, it could be called *SPW-2006 Example* in this paper.

4.1 SPW-2006 Example Problem

Extended from ISPW-6 Example, the *SPW-2006 Example* focuses on the various aspects that are affected by a change caused by a requirements change request. These aspects include not only the engineering process, but also the management process, support process, process improvement process, and so on. *SPW-2006 Example* extends ISPW-6 Example by:

- Expanding the problem to an integrated software development scenario in which *process*, *technology*, and *human* are all essential factors.
- Generalizing steps to *activities*; refining organizations to *actors*.
- Classifying steps into two categories, *component activities* that may occur step by step such as Review Design and *ongoing activities* that keep on going as Configuration Management
- Adding more steps/*activities* that may occur concurrently
- Adding more *actors* for expanding the organizational scope from development to the whole organization
- Extending constraints to *interactions*

In the following description, we use Italic font to differentiate the added or modified elements from those of ISPW-6 Example.

- ***Activities***
  - ***Component Activities***
    - ✓ ***Requirements Change Decision***
    - ✓ ***Technical Solution Decision***
    - ✓ ***Integration Test***
    - ✓ Schedule and Assign Tasks
    - ✓ Modify Design
    - ✓ Review Design
    - ✓ Modify Code
    - ✓ Modify Test Plans
    - ✓ Modify Unit Test Package
    - ✓ Test Unit
  - ***Ongoing Activities***
    - ✓ ***Configuration Management***
    - ✓ ***Cost Estimation***
    - ✓ ***Project Management (extended from "Monitor Progress" in ISPW-6 Example)***
    - ✓ ***Measurement and Improvement (including "Process Change" in ISPW-7 extension)***
    - ✓ ***Training***
    - ✓ ***Knowledge Management and Reuse***
- ***Actors***
  - ***SEPG***
  - CCB (Configuration Control Board)
    - ✓ ***SCM***

- Project Team
  - ✓ ***SQA (extended from "QA engineers" in ISPW-6 Example)***
  - ✓ ***Knowledge Engineer***
  - ✓ ***Estimation Expert***
  - ✓ ***Requirements Engineer***
  - ✓ Project manager
  - ✓ Design engineers
  - ✓ Software engineers
- ***User Representative***
- ***Trainer***

■ Artifacts
- Input + Source (***Artifact***, ***Actor***, or ***Activity***) + Physical communication mechanism
- Output + Destination (***Artifact***, ***Actor***, or ***Activity***) + Physical communication mechanism

■ ***Interactions (extended from "Constraints regarding step sequencing" in ISPW-6 Example)***
- ***Teamwork (as mentioned in ISPW-7 extensions)***
- ***Integration***

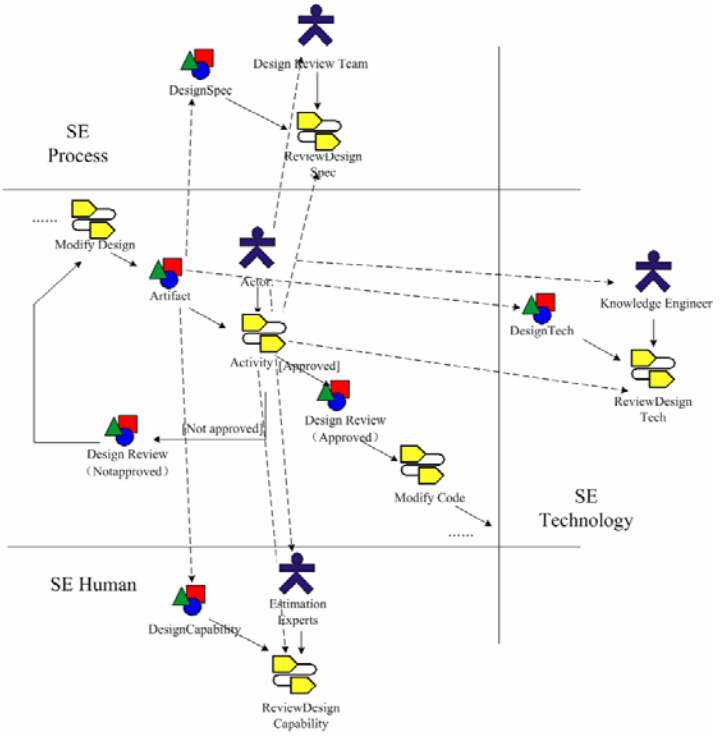

Fig. 7. Segment of SPW-2006 Example Represented by I-SPEM

Table 1. From ISPW-6 Example to *SPW-2006 Example*

	ISPW-6 Example	***SPW-2006 Example***
Actor	DesignReviewTeam	DesignReviewTeam
		KnowledgeEngineer
		EstimationExpert
Activity	ReviewDesign	ReviewDesignSpec
		ReviewDesignTech
		ReviewDesignCapability
Artifact	DesignSpec	DesignSpec
		DesignTech
		DesignCapability

In order to highlight the extension we made in *SPW-2006 Example*, comparing to ISPW-6 Example in Figure 1, we also use Review Design as the segment example in this section. Using *I-SPEM*, the problem is considered in the three dimensions of TRISO-Model, as shown in Figure 7. The *Actor*, *Activity* and *Artifact* are all expanded to groups of corresponding elements involved in the three dimensions. The mapping of these elements, from 1-D to 3-D, is illustrated in Table 1.

Using *CTM*, the dynamic semantics of the segment around the Review Design activity for the *SPW-2006 Example* is presented as follows:

$$SPW\_2006Part = ModifyDesign; ReviewDesign;$$
$$SKIP \lessdot approved \gtrdot SPW\_2006Part; ModifyCode$$

A design is firstly modified and then the modified design is reviewed. If the modified design is not approved, the design should be modified again. Otherwise the source code will be changed according to the approved design.

An activity in TRISO-Model is described from the three dimensions. The following three processes are used for modeling an activity:

$$\begin{array}{ll} ProcessActivity & \text{describing the actions taken out by the activity;} \\ HumanActivity & \text{describing the involved actors;} \\ TechnologyActivity & \text{describing the involved artifacts;} \end{array} \quad (4)$$

As for the Review Design activity, the processes in (4) can be correspondingly defined as:

$$\begin{aligned} ProcessReviewDesign &= rds : ReviewDesignSpec; rdt : ReviewDesignTech; \\ &\quad rdc : ReviewDeisignCapability \\ HumanReviewDesign &= drt : DesignReviewTeam; ke : KnowledgeEngineer; \\ &\quad ee : EstimatonExpert \\ ArtifactReviewDesign &= desspec : DesignSpec; dt : DesignTech; \\ &\quad dc : DesignCapability \end{aligned} \quad (5)$$

Firstly, the SE Process and SE Human dimensions are combined together through the *Management Integration*. We have,

$$ReviewDesignMI = ProcessReviewDesign \Theta_{MI:\{\{rds,drt\},\{rdt,ke\},\{rdc,ee\}\}} HumanReviewDesign$$

Then the $ArtifactsReviewDesign$ process integrates this above process through the *Data Integration* and the *Use Integration*. Thus the *Review Design* activity is modeled as:

$$ReviewDesign = ArtifactReviewDesign$$
$$\Theta_{UI:\{\{desspec,drt\},\{dt,ke\},\{dc,ee\}\}}\Theta_{DI:\{\{desspec,rds\},\{dt,rdt\},\{dc,rdc\}\}}ReviewDesignMI$$

It should be noted that the order of the application of the three external integrations does not matter.

4.2 Evaluations with the *SPW-2006 Example*

S.Fogle et al proposed six phases of a benchmarking project [39]: project initiation, planning, benchmarking partner identification, data collection, data analysis, and reporting. D.Card and D.Zubrow summarized three critical factors to success [40]: well-defined objectives, careful planning and cautious interpretation. The *SPW-2006 Example* benchmarking may be conducted according to the following three steps:

(1) Planning: decompose the benchmarked object into the corresponding or relative elements in *SPW-2006 Example*, based on its evaluation goals;
(2) Benchmarking: compare the decomposed elements with those in *SPW-2006 Example*;
(3) Evaluating: analyze the benchmarked object's similarities and differences with *SPW-2006 Example* and report the result.

As defined in section 4.1, the *SPW-2006 Example* benchmark includes 4 aspects, 31 elements. We use three levels of satisfactions to identify the current practice gaps to the *SPW-2006 Example* benchmark: *Not Satisfied* (*N*), *Partially Satisfied* (*P*) and *Fully Satisfied* (*F*). A benchmarked element at *N* or *P* level indicates a further software development process improvement opportunity.

Table 2 illustrates the *SPW-2006 Example* evaluation result to TRISO-Model. All the decomposed elements in TRISO-Model are *Fully Satisfied* (*F*) in comparison with the elements in the *SPW-2006 Example* benchmark. It shows that TRISO-Model is a good model to describe 3-D integrated software development processes.

Like the ISPW-6 Example, the *SPW-2006 Example* is originally designed for assessing the software process modeling approaches, particularly for the evaluation of an integrated software development process model, i.e., TRISO-Model. Furthermore, it expands the problem from one dimension of *process* to an integrated software development scenario in which three dimensions of *process*, *technology*, and *human* are all essential factors. Thus, it also may be used to evaluate software organizations, software projects and software development processes.

Table 2 also shows the evaluation results to other models for software development processes, CMM [41]/CMMI [10], ISO 9001[42], and SEPRM[43], with the *SPW-2006 Example*. In Table 2, many elements of the ISO 9001 standard is labeled with *Partially Satisfied* (*P*) or *Not Satisfied* (*N*) for the reason that the corresponding elements are just discussed in a broad sense. The difference between TRISO-Model and the other two ones is minor. However, from an analysis to the similarities and differences between each one-to-one element in the benchmarked object and *SPW-2006 Example*, a consensus conclusion should be reached that the CMM/CMMI is suitable for a software process integrated process, project and engineering management, but not suitable for detailed technological support, knowledge-based solution and cost estimation; ISO 9001 is suitable for a general process control, but not

Table 2. Evaluations with the SPW-2006 Example

SEPRM	ISO 9001	CMM/ CMMI	TRISO-Model				
F	*P*	*F*	*F*	***Requirements Change Decision***	***Component Activities***	***Activities***	SPW-2006 Example Benchmark
P	*P*	*P*	*F*	***Technical Solution Decision***			
F	*F*	*F*	*F*	***Integration Test***			
F	*F*	*F*	*F*	Schedule and Assign Tasks			
F	*P*	*F*	*F*	Modify Design			
F	*P*	*F*	*F*	Review Design			
F	*P*	*F*	*F*	Modify Code			
F	*P*	*F*	*F*	Modify Test Plans			
F	*P*	*F*	*F*	Modify Unit Test Package			
F	*P*	*F*	*F*	Test Unit			
F	*P*	*F*	*F*	***Configuration Management***	***Ongoing Activities***		
F	*N*	*F*	*F*	***Cost Estimation***			
F	*F*	*F*	*F*	***Project Management***			
F	*P*	*F*	*F*	***Measurement and Improvement***			
F	*F*	*F*	*F*	***Training***			
P	*N*	*P*	*F*	***Knowledge Management and Reuse***			
F	*N*	*F*	*F*	***SEPG***		***Actors***	
F	*P*	*F*	*F*	***SCM***	CCB		
F	*P*	*F*	*F*	***SQA***	Project Team		
N	*N*	*N*	*F*	***Knowledge Engineer***			
P	*N*	*P*	*F*	***Estimation Expert***			
F	*F*	*F*	*F*	***Requirements Engineer***			
F	*F*	*F*	*F*	Project manager			
F	*F*	*F*	*F*	Design engineers			
F	*F*	*F*	*F*	Software engineers			
F	*F*	*F*	*F*	***User Representative***			
F	*F*	*F*	*F*	***Trainer***			
F	*F*	*F*	*F*	Input + Source + physical communication mechanism		Artifacts	
F	*F*	*F*	*F*	Input + Destination + physical communication mechanism			
F	*P*	*F*	*F*	***Teamwork***		***Interactions***	
F	*P*	*F*	*F*	***Integration***			

Not Satisfied (N), Partially Satisfied (P), and *Fully Satisfied (F)*

suitable for specific software process management; SEPRM is a very good software engineering process reference model integrated 3 process subsystems of organization, development and management, but still lack of enough support to technology, knowledge and economy. TRISO-Model is a fully support reference model for integrated software development processes from the three most important dimensions of *process*, *technology*, and *human* naturally.

TRISO-Model has many unique features that are beneficial to the performance, analysis, and improvement of software processes. In TRISO-Model, the interrelationships among the elements of the software development process entities can be represented in *I-SPEM*, which includes more stereotypes and suits the convenience of modelers; and all the entities are uniformly described in their behavior patterns and are mapped onto *activity*, *artifact*, and *actor* in three dimensions through *CTM*, which guides the performance of development processes with rigorous operational semantics. New techniques for the analysis of software processes can be put forward based on the formalism. With the description of *artifacts*, the technical factors that transform the user requirement into the final product are covered in TRISO-Model. As human constitutes the major part of the cost of a project, various models for measurement and cost estimation can be integrated into the model through the modeling of *actor*.

In a simulated world, *SPW-2006 Example* benchmarking only adopts pass/fail strategy. Some parts of *SPW-2006 Example* may be changed to a quantitative way for real applications, e.g., "*Measurement and Improvement*" element was developed to an effective measurement method [44], which can be used to help identifying, analyzing, and solving the problems arising during the development processes.

In terms of the seven desiderata for successful benchmarks presented by Sim et al. in [23], the *SPW-2006 Example* fared very well as follows: (1) Accessibility: the *SPW-2006 Example* is an extended ISPW-6 Example and easily to be understood, to be found and to be used. (2) Affordability: people may use it to have an overall assessment to integrated software development processes. The costs are caused by human efforts and tools support, depending on how details the assessment needs to be. (3) Clarity: *SPW-2006 Example* is clear enough to describe software development processes through the elements in the three necessary aspects of *activities*, *actors*, *artifacts* and their interactions. (4) Relevance: it can be used to assess not only a general software development process, but also some specific software engineering processes like requirement engineering, software measurement. (5) Solvability: it is a good example to evaluate other software process models, and/or to evaluate software organizations, software projects and also software development processes, particularly 3-D integrated software development processes. (6) Portability: it is of course easy to be implemented at a variety of platforms. (7) Scalability: it is an extended version of ISPW-6 Example and definitely may be further extended to more complicated examples or even to a commercial product.

5 TRISO-Model Based Assessment and Improvement

The purpose of the assessment process is to efficiently find evidence of key process areas and identify areas for improvement [45]. The essential process activities are:

plan the assessment, distribute assessment material, prepare for assessment meeting, conduct assessment, make changes for improvement, and follow-up. The input to an assessment is the work item (project) to be scrutinized, the relevant checklists enumerating the types of key process areas to be identified, and other documents such as procedures and standards. The output of an assessment is firstly the log of the key process areas uncovered and secondly the areas for improvement and thirdly a summary report showing the score.

Several software process assessment models have been developed, such as CMM/CMMI, ISO 9001, ISO/IEC 15504 [46], and SEPRM.

5.1 TRISO-Model Based Assessments

There are two kinds of TRISO-Model based assessments: *TRISO-Model Qualitative Assessment* and *TRISO-Model Quantitative Assessment*.

The *TRISO-Model Qualitative Assessment* provides a checklist-based assessment method. It is also a kind of benchmark-based assessment. The benchmark is *SPW-2006 Example* in this paper. By comparing each element in the given software development process with the one in *SPW-2006 Example*, a pass/fail checklist will be given and the final assessment result will be made according to the pass/fail information. It is a very simple assessment methodology. It may be used to evaluate whether an assessed integrated software development process is well defined or not. But it is not helpful in step-by-step process improvement.

The *TRISO-Model Quantitative Assessment* provides a flexible software development process assessment method, based on the evaluation to integrated capability maturity levels. It may be written in a triplet as follows:

TRISO-Model Quantitative Assessment = (*PCM Level, TCM Level, HCM Level*)

where *PCM* represents *Process Capability Maturity*, *TCM* represents *Technology Capability Maturity* and *HCM* represents *Human Capability Maturity*. The *PCM Level*, *TCM Level* and *HCM Level* mean its process capability maturity level or status, technology capability maturity level or status, and human capability maturity level or status, respectively, in an integrated software development process for a software organization or a software project.

Each of capability maturities in the TRISO-Model three dimensions may be modeled as some available assessment model or a new assessment model. The integration of the three assessment models will be the TRISO-Model quantitative assessment model.

Based on CMM/CMMI and P-CMM for *PCM Level* and *HCM Level* respectively, this section presents a *TRISO-Model Quantitative Assessment Reference Model*. Accordingly, there are five defined maturity levels in *PCM Level*: *Initial* focuses on competent people and heroics *(1)*, *Repeatable* focuses on basic project management *(2)*, *Defined* focuses on process standardization *(3), Managed* focuses on quantitative management *(4)* and *Optimizing* focuses on continuous process improvement *(5)*; and five defined maturity levels in *HCM Level*: *Initial* initiates no processes *(1)*, *Repeatable* focuses on establishing basic workforce practices and eliminating problems that hinder work performance *(2)*, *Defined* addresses organizational issues, as the organization tailors its defined workforce practices to the core competencies required by its

business environment *(3), Managed* focuses on building competency-based teams and establishing a quantitative understanding of trends in the development of knowledge and skills and in the alignment of performance across different levels of the organization *(4)* and *Optimizing* covers issues that both the organization and individuals must address in implementing continuous improvements in their capability *(5)*.

Here we define the maturity levels in *TCM Level* on our own, also five levels to match *PCM Level* and *HCM Level*: *Initial* initiates software development *(1)*, *Repeatable* focuses on establishing necessary domain knowledge support *(2)*, *Defined* addresses technology standardization and tool support *(3), Managed* emphasizes technology innovation and management *(4)* and *Optimizing* aims at a technological leadership and continuous technology improvement *(5)*.

Thus, an assessment result based on *TRISO-Model Quantitative Assessment Reference Model* will be the three numbers combination of the triplet between *(1, 1, 1)* and *(5, 5, 5)*. For an example, an assessment result *(4, 3, 4)* means that the assessed software development process achieved an integrated level *(4, 3, 4),* with *Managed* process capability maturity level, *Defined* technology capability maturity level and *Managed* human capability maturity level, respectively. It performed a quantitative process management, used development tools support and possessed a good qualified team.

Table 3 shows a TRISO-Model based assessment form. The *TRISO-Model Quantitative Assessment* evaluates the assessed software development process from the three dimensions of *Process Capability Maturity (PCM) Level (1-5), Technology Capability Maturity (PCM) Level (1-5)* and *Human Capability Maturity (PCM) Level (1-5)*, by assessing the six integrations as shown in Figure 2. It can be conducted through three steps as follows.

Table 3. TRISO-Model Based Assessment Form

	TRISO-Model Quantitative Assessment		
	PCM Level (1-5)	*TCM Level* (1-5)	*HCM Level* (1-5)
Process Integration		—	—
Development Integration	—		—
Service Integration	—	—	
Data Integration			—
Management Integration		—	
Use Integration	—		
Assessment Result			

Firstly, it assesses the three internal integrations of *Process Integration, Development Integration* and *Service Integration*.

Secondly, it assesses the three external integrations of *Data Integration, Management Integration* and *Use Integration*. To assess *Data Integration*, factors in both process dimension and technology dimension have to be taken into account; and it is similar with *Management Integration* and *Use Integration*.

Finally, it accounts the assessment result to each dimensional capability maturity level, i.e., achieving the result of *Process Capability Maturity (PCM) Level* by

accounting the three assessment scores of the internal *Process Integration* and the two relative external *Data Integration* and *Management Integration*; it is the same with the *Technology Capability Maturity (TCM) Level* and the *Human Capability Maturity (HCM) Level.*

From the viewpoint of model framework structures, Wang and Bryany observe the current assessment methods as three types [47]: (1) Checklist-based assessment, i.e., a software process assessment method that is based on a pass/fail checklist for each practice and process specified in a process model. The ISO 9001 model provides a checklist-based assessment method. (2) 1-D process-based assessment, i.e., a software process assessment method that determines a software development organization's capability from a set of processes in a single process dimension. CMM is an example of 1-D assessment models. (3) 2-D process-based assessment, i.e., a software process assessment method that employs both process and capability dimensions in a process model, and derives process capability by evaluating the process model against the capability model. ISO/IEC 15504 and SEPRM are examples of 2-D assessment models.

As the above discussions, the *TRISO-Model Qualitative Assessment* is a checklist-based assessment method; and the *TRISO-Model Quantitative Assessment* presents a new type of assessment method, i.e., a kind of "3-D" process-based assessment from the three dimensions of *process*, capability (*human* in TRISO-Model) and *technology*.

5.2 Improving 3D Integrated Software Development Processes

There are three key categories of philosophies underpinning software process improvement [47]: (1) Goal-oriented process improvement, i.e., a software process improvement approach by which process system capability is improved by moving towards a predefined goal, usually a specific process capability level. It is simple and the most widely adopted software engineering philosophy. ISO 9001 provides a pass/fail goal; CMM, ISO/IEC 15504, and SEPRM provide a 5/6-level capability goal. (2) Benchmark-based process improvement, i.e., a software process improvement approach by which process system capability is improved by moving towards an optimum combined profile according to software engineering process benchmarks, rather than a maximum capability level. It presents empirical indications of process attributes. This approach provides an organization with sufficient margins of competence in every process. (3) Continuous process improvement, i.e., a software process improvement approach by which process system capability is required to be improved all the time, and toward ever higher capability levels. Using this approach, software process improvement is a continuous, spiral-like procedure and there is no end to process optimization. It provides a basis for sustainable long-term strategic planning. The Deming Circle, plan-do-check-act, is a typical component of this philosophy.

Though there is a criticism that the goals for improvement are not explicitly stated in continuous process improvement philosophy and top management has to make clear the current goal, as well as the short, middle, and long-term ones, TRISO-Model is principally a continuous improvement approach with some staged goals or benchmarks to provide more precise assessment results.

There are three basic software process improvement methods [47]: (1) Assessment-based improvement, i.e., a software process improvement method in which a process system can be improved by basing its performance and capability profile on either a

model-based or a standard-based assessment. Using this approach, the processes inherent in a software development organization are improved, according to a process system model with step-by-step suggestions like CMM, or a standardized process system model like ISO/IEC 15504. (2) Benchmark-based improvement, i.e., a software process improvement method in which a process system can be improved by basing its performance and capability profile on a benchmark-based assessment. Using this approach, the processes inherent in a software development organization are improved according to a set of process benchmarks. SEPRM is a benchmarked model, which provides an optimized and economical process improvement solution. (3) Integrated improvement, i.e., a combined model-based and benchmark-based software process improvement method in which a process system can be improved by basing its performance and capability profile on a integrated model-based and benchmark-based assessment. Using this approach, the processes inherent in a software development organization are improved according to a benchmarked process system model. SEPRM is designed to support integrated model- and benchmark-based process improvement, which inherits the advantages of both absolute and relative software process improvement methods.

TRISO-Model basically is a model-based process improvement methodology, but also it may introduce some benchmark-based improvement, and then to be an integrated improvement.

The conventional goal-based process assessment and improvement technologies have been widely accepted. However, its philosophy of "the higher the better" has been questioned in practice [24]. The determination of target capability levels for specific organization tends to be virtual, infeasible, and sometimes overshoot. Benchmark-based process assessment and improvement supports the philosophy of "the smaller the advantage, the better". CMMI continuous representation offers a flexible approach to process improvement [10]. An organization may choose to improve the performance of a single process-related trouble spot, or it can work on several areas that are closely aligned to the organization's business objectives; and to improve different processes at different rates.

TRISO-Model presents a new integrated improvement method. It adopts the philosophy of "the smaller the integrated goal, the better". The target capability maturity levels of given software development processes will be set relative to the next integrated goal, rather than to the virtually highest level as in a goal-based process assessment and improvement, or to the benchmarks of the software industry as in a benchmark-based process assessment and improvement.

For the given assessment result example *(4, 3, 4)* in section 5.1, it is a very good software organization if it focuses on international outsourcing. However, when it would like to evolve into a software product vendor, i.e., developing its own innovative technology or product, it has to improve its technology capability maturity firstly.

TRISO-Model is not only suitable for process improvement from process scope to project and organization scopes because it may provide precise measurement for every process at all the capability levels like ISO/IEC 15504 and SEPRM, but also very important for process improvement in technology scopes because it may provide advanced software technologies support to either software development processes for higher capability levels, or the project or organization's schedule/budget control.

As a simulation-based research up to now, also, more work is needed to mature the overall approach in order to make it a reliable, cheap, and easy-to-apply support tool for decision makers in software process improvement programmes, as Pfahl and Birk indicated in [48].

6 Conclusions and Future Work

TRISO-Model is developed to improve software development practices in the current complex and dynamic environment by describing and managing the elements contributing to project success in three interactive dimensions, i.e. SE Human, SE Process, SE Technology, and their integrations. With TRISO-Model, various aspects of projects, such as people, tools, and processes, can be modeled and managed systematically. The static semantics and the dynamic semantics of TRISO Model are specified by an extension of SPEM, *I-SPEM*; and by an extension of CSP, *CTM,* respectively. New techniques for the analysis of software development processes can be put forward based on the formalism.

In order to simulate and evaluate the modeling ability of TRISO-Model, we create a new process benchmark, *SPW-2006 Example*, by extending the ISPW-6 Example. Unlike the process-centered ISPW-6 Example, the *SPW-2006 Example* is not oriented to any specific single aspect of software development but it incorporates more aspects by adding more elements and interactions. It may be used to evaluate other software process models, and/or to evaluate software organizations, software projects and also software development processes, particularly 3-D integrated software development processes. The evaluation shows that the TRISO-Model approach is effective in modeling and managing different aspects and their complex interactions in today's software development.

With the *SPW-2006 Example* and its evolution for quantitative evaluation to 3-D integrated software development processes, we present two kinds of TRISO-Model based assessments: *TRISO-Model Qualitative Assessment* and *TRISO-Model Quantitative Assessment*. It enables a new integrated improvement method for software development processes.

The *TRISO-Model Qualitative Assessment* provides a checklist-based assessment method. It may be used to evaluate whether an assessed integrated software development process is well defined or not, based on the *SPW-2006 Example* benchmark. The *TRISO-Model Quantitative Assessment* provides a flexible software development process assessment method, based on the evaluation to integrated capability maturity levels of *process*, *technology* and *human*. It may be used in an integrated environment, as a continuous improvement approach with some staged goals or benchmarks to provide more precise assessment results.

Last, but not least, from the viewpoint of end-users (consumers) or investors (producers), software is always viewed as a true investment, not just a development activity, and therefore is evaluated in terms of the value created rather than only the functionality delivered. Thus, all the 3-D software development process assessment and improvement should be mapped into the return on investment (ROI) factor finally [49]. It is of course the next direction for further research.

Acknowledgements

The presentation was supported partly by the National Natural Science Foundation of China (Grant Number: 60573082). Also, I appreciate all the help offered by my colleagues (particularly to Qing Wang, Yongji Wang and Chen Zhao) and students (specially to Feng Yuan, Qiusong Yang, Jizhe Wang and Da Yang) in the Lab for Internet Software Technologies, Institute of Software at Chinese Academy of Sciences.

References

1. M.Li: Expanding the Horizons of Software Development Processes: A 3-D Integrated Methodology. In: Mingshu Li, Barry Boehm and Leon J. Osterweil (eds.), Unifying the Software Process Spectrum, Software Process Workshop (SPW2005; May 25-27, 2005). LNCS 3840, Springer-Verlag (2005) 54-67
2. A.Wise et al.: Using Little-JIL to Coordinate Agents in Software Engineering. In: Proc. of the Automated Software Engineering Conf. (2000) 155-163
3. N.Belkhatir et al.: Adele/Tempo: An Environment to Support Process Modeling and Enaction. In: A.Finkelstein et al., Software Process Modelling and Technology, John Wiley & Sons, Inc. (1994) 187-217
4. B.W.Boehm et al.: Software Cost Estimation with COCOMO II. Prentice-Hall (2000)
5. M.Ruhe, R.Jeffery and I.Wieczorek: Cost Estimation for Web Applications. In: Proc. of 25th Int. Conf. on Software Engineering (ICSE 25) (2003) 270-279
6. B.Boehm and A.Jain: An Initial Theory of Value-Based Software Engineering. In: A. Aurum et al.(eds.): Value-Based Software Engineering. Springer-Verlag (2005)
7. W.S.Humphrey: Introduction to the Personal Software Process. Addison-Wesley (1997)
8. B.Curtis et al.: People Capability Maturity Model. Addison-Wesley (2001)
9. D.Reifer: Ten Deadly Risks in Internet and Intranet Software Development. IEEE Software, March/April (2002) 12-14
10. M.B.Chrissis et al.: CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley (2003)
11. B.Boehm and D.Port: Balancing Discipline and Flexibility with the Spiral Model and MBASE. Crosstalk, Vol.11(12) (2001) 23-28
12. J.Estublier: Software are Processes Too. In: Mingshu Li, Barry Boehm and Leon J. Osterweil (eds.), Unifying the Software Process Spectrum, Software Process Workshop (SPW2005; May 25-27, 2005). LNCS 3840, Springer-Verlag (2005) 25-34
13. H.D.Rombach: Integrated Software Process & Product Lines. In: Mingshu Li, Barry Boehm and Leon J. Osterweil (eds.), Unifying the Software Process Spectrum, Software Process Workshop (SPW2005; May 25-27, 2005). LNCS 3840, Springer-Verlag (2005) 83-90
14. L.J.Osterweil: Integrating Microprocess and Macroprocess Software Research. In: Mingshu Li, Barry Boehm and Leon J. Osterweil (eds.), Unifying the Software Process Spectrum, Software Process Workshop (SPW2005; May 25-27, 2005). LNCS 3840, Springer-Verlag (2005) 68-74
15. B.Warboys: Active Models: A Possible Approach to the Integration of Objective and Subjective Process Models. In: Mingshu Li, Barry Boehm and Leon J. Osterweil (eds.), Unifying the Software Process Spectrum, Software Process Workshop (SPW2005; May 25-27, 2005). LNCS 3840, Springer-Verlag (2005) 100-107
16. M.Ruiz et al.: A Dynamic Integrated Framework for Software Process Improvement. Software Quality Journal, 10 (2002) 181-194

17. M.I.Kellner, R.J.Madachy and D.M.Raffo: Software Process Simulation Modeling: Why? What? How? Journal of Systems and Software, 46 (2/3) (1999) 91-105
18. M.Ruiz et al.: Using Dynamic Modeling and Simulation to Improve the COTS Software Process. In: F.Bomarius and H.Iida (eds.), PROFES 2004. LNCS 3009, Springer-Verlag (2004) 568-581
19. P.Donzelli and G.Iazeolla: Hybrid Simulation Modeling of the Software Process. Journal of Systems and Software, 59 (2001) 227-235
20. R.H.Martin and D.Raffo: A Model of the Software Development Process Using both Continuous and Discrete Models. Software Process Improvement and Practice, 5 (2000) 147-157
21. H.Neu and U.Becker-Kornstaedt: Learning and Understanding a Software Process through Simulation of its Underlying Model. In: S.Henninger and F.Maurer (eds.), LSO 2002. LNCS 2640, Springer-Verlag (2002) 81-93
22. N.David et al.: Towards an Emergence-Driven Software Process for Agent-Based Simulation. In: J.S.Sichman et al. (eds.), MABS 2002. LNAI 2581, Springer-Verlag (2003) 89-104
23. S.E.Sim et al.: Using Benchmarking to Advance Research: A Challenge to Software Engineering. In Proc. of the 25th Int. Conf. on Software Engineering (2003) 74-83
24. Y.Wang and G.King: A New Approach to Benchmark-Based Process Improvement. In: Proc. of European Software Process Improvement 2000 (2000) 140-149
25. M.I.Kellner et al.: ISPW-6 Software Process Example. In Proc. of the First Int. Conf. on Software Process. IEEE Computer Society Press (1991) 176-186
26. D.Avrilionis et al.: OPSIS: A View Mechanism for Software Processes which Supports their Evolution and Reuse. In Proc. of the 18th Int. Conf. on Software Engineering (1996) 38-47
27. C.M.Lot and H.D.Rombach: A MVP-L1 Solution for the Software Process Modeling Problem. In Proc. of 6th Int. Software Process Workshop (ISPW 6) (1990)
28. G.Junkermann et al.: Merlin: Supporting Cooperation in Software Development through a Knowledge-based Environment. In Software Process Modelling and Technology. John Wiley & Sons, Inc. (1994) 103-127
29. N.Belkhatir, J. Estublier and W.L.Melo: Software Process Modeling in Adele: the ISPW-7 Example. In: Proc. of the 7th International Software Process Workshop (1991) 48 -50
30. F.P.Brooks: No Silver Bullet: Essence and Accidents of Software Engineering. Computer, Vol.20(4) (1987) 10-19
31. OMG: Software Process Engineering Metamodel Specification, Version 1.1 (formal/2005-01-06). (2005) (http://www.omg.org)
32. P.Kruchten: A Process Engineering MetaModel. (2001) (http://www.forsoft.de/zen/sdpp02/papers/Kruc01.pdf)
33. C.Kobryn: UML 2001: A Standardization Odyssey. Communications of the ACM, 42(10) (1999) 29-37
34. OMG: MOF Core Specification, Version2.0 (ptc/2003-10-04). (2003) (http://www.omg.org)
35. C.A.R.Hoare: Communicating Sequential Processes. Prentice Hall International (1985)
36. R.M.Greenwood: Using CSP and System Dynamics as Process Engineering Tools. In Proc. of the 2nd European Workshop on Process Technology (Trondheim, Norway, Sept. 7-8, 1992). Springer-Verlag (1992) 138-145
37. K.Yasumoto et al.: Software Process Description Using LOTOS and its Enaction. In Proc. of the 16th Int. Conf. on Software Engineering (1994) 169-178
38. A.W.Roscoe: The Theory and Practice of Concurrency. Prentice-Hall Pearson (2005)

39. S.Fogle et al.: The Benchmarking Process: One Team's Experience. IEEE Software, Septeber/October (2001) 40-47
40. D.Card and D.Zubrow: Benchmarking Software Organizations. IEEE Software, Septeber/October (2001) 16-18
41. CMU SEI: The Capability Maturity Model Guidelines for Improving the Software Process. Addison-Wesley, Pearson Education (1994)
42. International Standard: ISO 9001 Quality Management System – Requirements (2000)
43. V.Chiew and Y.Wang: Software Engineering Process Benchmarking. In: M.Oivo and S.Komi-Sirvio (eds.), PROFES 2002. LNCS 2559, Springer-Verlag (2002) 519-531
44. Q.Wang and M.Li: Measuring and Improving Software Process in China. In: Proc. of 2005 International Symposium on Empirical Software Engineering (ISESE) (2005) 183-192
45. E.Gray et al.: An Incremental Approach to Software Process Assessment and Improvement. Software Quality Journal, 13 (2005) 7-16
46. International Standard: ISO/IEC 15504 - 1-9, Software Process Assessment – Parts 1-9 (2000)
47. Y.Wang and A.Bryany: Process-Based Software Engineering: Building the Infrastructures. Annals of Software Engineering, 14 (2002) 9-37
48. D.Pfahl and A.Birk, Using Simulation to Visualise and Analyse Product-Process Dependencies in Software Development Projects. In: F.Bomarius and M.Oivo (eds.), PROFES 2000. LNCS 1840, Springer-Verlag (2000) 88-102
49. K.E.Emam: The ROI from Software Quality. Auerbach Publications, Taylors & Francis Group (2005)