An Approach to Handling Software Process Deviations

Rui Zhu, Fei Dai, Qi Mo, Yong Yu, Leilei Lin, and Tong Li

***Abstract*—Since software processes are permanently dynamic evolving during software process enactment, the problems on software process deviation are increasingly received more attention. In process-centered Software Engineering Environments (PSEE), the enacted software process model is always inconsistence with the observed process obtained by PSEE. In order to address this problem, this paper presents a framework which structures the field of process deviation analysis and identities, a Process Behavior Space Expression based on process algebra and judgment rules to decide whether handle the deviation or not. This approach can effectively find and handle the prevalent deviation problems in the software process implementation, through dealing with the deviation so as to improve the software process, and ultimately promote the quality of software products.**

***Index Terms*—Software process deviation, process algebra, process behavior space expression, deviation judgment rules, deviation handling.**

I. Introduction

With in-depth knowledge on the PSEE (Process-centered Software Engineering Environment), researchers and practitioners have realized that there have been always a certain deviation and inconsistency between the observed software process actually and the process model. The deviation and inconsistency mainly are result from 1) incompletion of process model naturally and 2) human existence. The modelers cannot obtain the whole requirements and the modeling tools only architect part of the information about the requirements lead to an incomplete process model. The human existence, an inevitable critical factor in software engineering, generates the uncertainty of process enactment. This character reflects the nature of a creative activity such as software development, where consistency is the exception and not the rule [1].

The objective of the work presented in this paper is to solve the deviation of software process so as to enhance the guidance of PSEE. The Addressed processes are those of organizations where the software process corresponds to "The Defined Level" of maturity according to the Capability Maturity Model [2]. The PSEE was proposed by the current researchers, who use the process-driven development, to manage the software development process better. Based on the Process Algebra, a Process Behavior Space Expression (PBSE) is proposed to detect and manage the process deviation. First of all, to describe the enacting software process, we use the Evolution Process Meta-Model (EPMM) [3], which is proposed by one of the authors to model the software process efficiently. The EPMM is divided into four levels: global, process, activity and task. In process level, it utilizes the classic condition/transition Petri net for modelling, through this meta-model we can describe the concurrency and iteration of the software evaluation process. Secondly, there are two models, the enacting process model (EPM) and the observed process model (OPM) which are described by the EPMM, have been proposed. In order to detect the deviation of the process, the EPM and OPM are compared. If there exist a deviation, the PBSE would detect, and people determine whether to handle the deviation from the Action Deviation Judgment Law.

The paper is organized as follows: Section II introduces some related work, mainly about the concepts of Software Process Deviation and a diagnosis Architecture of Process Deviation Based on EPMM. In Section III, we mainly discuss the way to judge whether to handle the deviation from the Behavior Deviation Judgment Rules. A case study will be given in Section IV. And finally, conclusions and further works are outlined in Section V.

II. Related Work

In 1987, Lean Osterweil puts forward a famous assertion "The Software Processes are Software Too". In the reference 4, he explained the nature of the process software in detail, and indicated that the Software Processes can be enacted like software. Nowadays, it is widely accepted that the quality of a software product is largely dependent on the development process used [2], [5]. For the software process improvement research, it mainly divided into two categories [6]: The first category is the software process assessment and improvement model which is concerned with the software industry and represented by the Capability Maturity Model Integration. The second category is the software process modeling followed by the academic. Through a specific method to abstract, express and analysis the software process so as to increase the understanding of the process, it guides the actual software development activities by direct or indirect way [7]. The research on the software process modeling method centers on the Process Modeling Language and PSEE [6]. However, with in-depth knowledge on the PSEE, researchers and practitioners have realized that a process model being enacted must be flexible enough to allow process changes to face unexpected situations [8]. Consequently, the actual observed process model deviates from the enacting process model.

Manuscript received July 12, 2014; revised September 16, 2014. This work has been supported by the National Science Foundation of China under Grant No. 60963007, 61262024, 61462091, 61462095 and 61262025, by the Key Subject Foundation of School of Software of Yunnan University and the Open Foundation of Key Laboratory in Software Engineering of Yunnan Province under Grant No. 2010KS01, by the Yunnan Province Science Foundation of China under Grant No. 2012FD005. Supported by the Sixth Yunnan University graduate research funded projects in 2013(No. ynuy75).

Rui Zhu and Tong Li are with Yunnan University, Kunming, Yunnan, China (e-mail: ray0209055@gmail.com).

A. Overview

The current research on the Software Process Deviation mainly concentrated on the deviation handling or the PSEE of the ability to tolerate the deviation. At the same time, we haven't only summarized the research results of Software Process Deviation, but also the research progress of the business flow. This mainly because there is a mapping relationship between the business workflow construction methods and software process modeling methods, and researchers have learned from each other a lot. In the reference 9, Cugola proposed a PSEE based on an artifact-oriented language PROSYT to deal with the problem of managing unforeseen situations that require deviations from the process model during enactment in the context of the PROSYT [9]. Pohl etc. put forward an integrated PSEE, in which a protocol has been joined between the process virtual execution and process practical enactment so as to reduce the deviation [10]. Mohammed etc. presents an original approach to software process enactment evolution based on dynamic adaptation of processes and tolerance of deviations [8]. Li etc. indicate that in recent years the researchers in the field of software process modeling have made a lot of useful exploration to address the problem [7]. The research focuses mainly center on the EPMM supporting for process evolution and deviation tolerance, the validation and analysis to software process (including the syntax checking to process model, semantic correctness analysis, matching and simulation), and integrated software process model etc. Ho-Won Jung and Dennis R. Goldenson provides basic proposition of process assessment models to support proposition by testing the hypothesis that higher process maturity is negatively associated with schedule deviation in software maintenance, and investigate whether two process context factors (organizational size and geographical region) modify the relationship between process maturity and schedule deviation by using a moderator testing method [11].

Throughout the study states, current researchers often consider how to deal with the process deviation and researching on the software process model of deviation handling. However, how to detect and find the deviation, the most important part of process deviation is rarely discussed, and it is also one of this paper starting points.

B. Software Process Deviation

Before the deviation discussed, some concepts should be given. The term *actual process* denotes the actual software process as it is performed in the real world [8]. The actual process is carried out implicitly in the organization business, and people in order to reuse or visualize it, build the process model. Consequently, the term *enacting process model* describes the actual process in an organization business, which is an instantiation of the software process model in fact. The enacting process model is built by people to describe the actual process, and it is an abstract of the actual process with strong human subjective intention. The modelers hope the whole development process is implemented in accordance with the model established, and do not want to take place the deviation. However, the deviation is not necessarily a mistake. The enacting process model may not be visible, because there may be not a process model in the development group. But in this paper, we only discuss the development organization possesses a process model to guide their development. That is to say, we only discuss the situation that the organization has a PSEE and people should model the process before the software development （Or model driven development, MDD）. And in this situation, the enactment process model is a mapping of the actual process. On the other hand, after the actual process is enacted, the observed process will be obtained by PSEE. The term *observed process* denotes the partial view of actual process owned by the PSEE [8]. The term *observed process model* denote a process model is obtained according to the process enacting record in PSEE utilizing the process mining technology. A deviation is an inconsistency between the SPM and the execution as observed by the PSEE [12]. Mohammed etc. argue the Software Process Deviation is different from the inconsistency, deviation is an action which is not described in the predefined process or violates some of the constraints, and the deviation can be divided into actual process deviation, observed process deviation, environment deviation and mining deviation [8].

Based on the analyzing system view of Process Algebra, we argue that the deviation denotes the difference between the actions of the system, and the inconsistence indicates the difference between states of the system. Furthermore, the differences between behaviors are more important than the states. And the deviation can be divided into four forms: actual process deviation, observed process deviation, environment deviation and mining deviation.

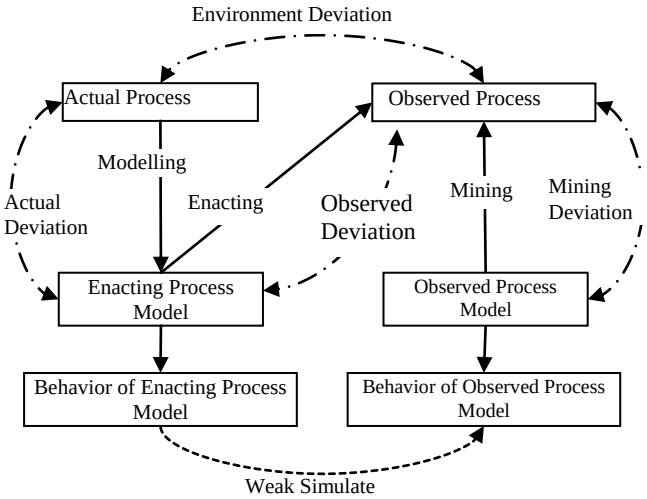

Fig. 1. The relationship between processes and models in SPD.

The relation between the actual software process, the enacting process model, the observed process and the observed process model is shown in Fig. 1.

In this paper, we mainly discuss the observed process deviation. Generally, the behavior of the enacting process model is richer than observed process, for those not taking place the observed process deviation, because in this situation the PSEE only enacted some actions which had been defined in the process model. And in order to illustrate the proposition better, we give software process definitions firstly.

Definition 1 [3]: A 4-tuple $\Sigma=(C, A, F, M)$ is called a software process system where

1) (C, A, F) is a net without isolating elements, $A \cup C \neq \Phi$;
2) C is a finite set of conditions; $\forall c \in C$ is called a condition;
3) A is a finite set of activities; $\forall a \in A$ is called an activity; the occurrence of a is called that a is executed or that a fires;
4) $F \subseteq C \times A \cup A \times C$, F denotes the flow relationship;
5) $M \subseteq 2^C$ is called the case class of $\sum$. 2^C denotes the power set of C;
6) $\forall a \in A, \exists m \in M$, such that a has concession in m.

The OPM and the EPM both can be described by EPMM. In order to research the Observed process deviation, we should firstly discuss a phenomenon that is a relationship between the observed process and the enacting process model.

Axiom 1: For un-deviated software process, the behavior of the EPM can weak simulate the OPM.

C. Diagnosis Architecture on Process Deviation

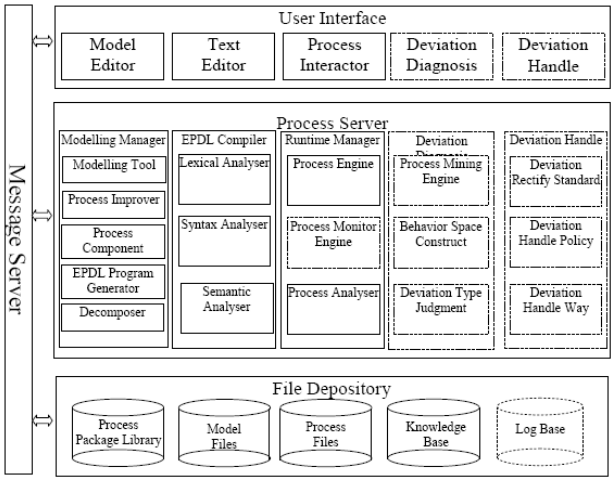

Fig. 2. The diagnosis architecture of software process deviation.

With the above sections researched and analyzed, the diagnosis Architecture of Process Deviation Based on EPMM will be given. In Fig. 2, the full lines denote the existing module in EPMM, and the dotted lines denote the modules what are added for research on the process deviation. The added modules are mainly divided into two parts: Deviation Diagnosis Engine and Deviation Handle Engine. The Process Monitor Engine is added into the Deviation Diagnosis Engine so as to analysis the Log Base, and then the OPM is mined using the process mining technology. The PSEE can use the behavior to construct aiming at the OPM that have been mined, and then the process deviation can be classified. When the deviation is detected, the deviation is submitted into the Deviation Handle Engine to handle. Software Process Behavior Modelling

In order to describe the software process behaviors, we should model them. A process is of some behaviors, and these behaviors are intertwined and interact with each other. Not only hierarchy, but also structural relationships among them. These whole behaviors of software process constitute their behavior space. In the theory of process algebra, process is an abstract between the system and external environment, and this is the behavior model of the software process. Behaviors consist of transitions, which can be enabled to make the system change from one state to another.

Definition 2: Transition relationship is a 3-tuple $p = (p, a, p')$, which can be noted to $p \xrightarrow{a} p'$ where

1) $p, p' \in P$, P is the set of process states;
2) a is an action.

As we know, software process model $p=(N, M_0)$, where N denotes the structure of Petri net, M_0 is called the initial marking of p. If we regard a process in process theory, a case in software process model is equivalent to a state of the process. When an activity A in software process model is fired to make software N changed from M_0 to M_1, we can use transition relationship to describe it to $(N, M_0) \xrightarrow{a} (N, M_1)$.

Definition 3: A software process behavior space denotes that all actions of software process would happen and the partial ordering relations among these actions. We regard a formal description of process behavior space as software process behavior space expression.

Because this paper mainly discusses behavior deviation in software process and the most important thing in behavior deviation is the behavior execution order, a Process Behavior Space Expression (shorted for PBSE) is put forward based on the process algebra. The grammar of PBSE defined using BNF as follows:

PBSE:: = <activity> | PBSE.PBSE | PBSE+PBSE | (PBSE|PBSE) | !PBSE | new <activity> PBSE

We define '.' as order operator to describe the meaning of execution by order, and it can be omit when the name of process won't produce misunderstanding, define '+' as choice operator to express the meaning of selection execution among processes, define '|' as concurrent operator to convey the meaning of concurrent execution among processes, define '!' as repetition operator to convey the meaning of repeated execution (repetitions can be 0), define 'new' as limit operator to convey the meaning of the name of activity limited in a behavior space expression, and '*<activity>*' is defined as the name of an activity in the software process model. In addition, a special activity δ is defined to denote the end of the identifier, and can be omitted.

In order to describe the every step of software process execution, we take consider to join he information of activity execution record into the PBSE expression. The transition rules of PBSE will be given as follows:

Definition 4 (Transition rules):

1) If $PBSE_{[\#]}=\pi.PBSE' \wedge \exists PBSE \xrightarrow{\pi} PBSE' \Rightarrow PBSE_{[\#\pi]}=PBSE'$.
2) If $PBSE_{[\#]}=(\pi+\tau).PBSE' \wedge \exists PBSE \xrightarrow{\pi \text{ or } \tau} PBSE' \Rightarrow PBSE_{[\#\pi]}=PBSE'$ *or* $PBSE_{[\#\tau]}=PBSE'$.
3) If $PBSE_{[\#]}=(\pi \mid \tau).PBSE' \wedge \exists PBSE \xrightarrow{\pi,\tau \wedge \pi >> \tau} PBSE' \Rightarrow PBSE_{[\#\pi\tau]}=PBSE'$, ">>" Denotes activity occurs antecedently.
4) If $PBSE_{[\#]}=!(\pi.\tau.)\varphi.PBSE' \wedge \exists PBSE \xrightarrow{\pi} PBSE' \Rightarrow PBSE_{[\#\pi]}=\tau.!(\pi.\tau.)\varphi.PBSE'$.
5) If $PBSE_{[\#]}=!(\pi.\tau.)\varphi.PBSE' \wedge \exists PBSE \xrightarrow{\varphi} PBSE' \Rightarrow PBSE_{[\#\varphi]}= PBSE'$.

III. Software Process Deviation Handling

In the previous chapter, how to produce the PBSE is

mainly introduced. The ultimate purpose of deviation research is to handle the deviation. Generally speaking, the more deviations are foreseen by the process supporting system, the more function it has. However, not all deviations can be foreseen, so when there are unforeseen deviation being found, it should be researched how to distinguish what deviations should be handled or unhandled. Since those deviations that need to be handled, it should be researched that the deviation handling policy would be adopted. This chapter takes the deviation handling as the start point, drives for problem to offer a solution for deviation handling.

A. Deviation Judgment Rules

In this paper, we mainly discuss observed process deviation, and we know that it is an action performed within the PSEE that is not reflected in the process model. There are three problems need to be solved. Problem 1: the standard of rectifying deviation; problem 2: the criterion of handling deviation; problem 3: how to solve the deviations according to different situations. For the first problem, the standard of rectifying deviation denotes that a process has taken place a deviation, and we need to judge whether it should be handled or not. As we know, there are some deviations need to be rectified, but some do not need. The deviations of software process are not all mistaken. For the second problem, the criterion of handling deviation denotes that if we would handle the deviation, what criterion will be accepted should be considered. In other words, we should figure out that the actual process deviated from model or model deviated from actual process. For the third problem, we should clear and definite how to solve the deviations according to different situations after confirming the criterion of handling deviation. This chapter tries to put forward a standard of rectifying deviation based on the target. The process target will be defined firstly in this solution, and then through the execution of process we will describe formally the products of process execution. At last, we will judge whether we need to rectify a deviation according to the comparison of the target description with the process products description.

Definition 5 (Process Target):

Target T is a 4-tuple, $T = (I, D, S, A)$, where I denotes the sequence number of target, such as 1,2,3…; D denotes description set of target T, and every description is defined by First Order Predicate; S denotes the state of T, which is used to describe that the target have been finished or unfinished, and it is a Boolean type (True or False); A denotes the activities set related to the target.

Definition 6 (Process Goal Set):

If the software process $P=(C, A, F, M_0)$, the process Goal set $PG = \{T | \forall_{T \in G \wedge p \in P} T.a \in p.A\}$, where T is a target, G is the set of all targets. In other words, the Process Goal Set denotes that those targets are corresponding to process P in the set of all targets.

In reference [13], Jacques has defined the software process as follows:

Definition 7 (Software Process):

A set of partially ordered process steps, with sets of related artifacts, human and computerized resources, organizational structures and constraints, intended to produce and maintain the requested software deliverables.

Through definition 7, we can know that software process is a set of partially order activities, and the inputs are transformed into outputs by these activities. The outputs are called "Products". So next, the definition of process outputs will be given as follows:

Definition 8 (Process outputs):

The outputs of the process are the set of products, and the description of products is called the software process description $POD = \{o | \text{the all product descriptions of } P\}$, where P denotes the software process.

After giving the definition 8, we can illustrate the standard of rectifying deviation: whether the Software Process Target Set is harmonious with Process Outputs or not. If they are harmonious, we maintain that even though there may be deviations in the enactment of software process, we can tolerate them. However, if the Software Process Target Set is not harmonious with Process Outputs, we consider the deviation must be handled. We define the need to rectify as ND, and the formal description of the standard of need to rectify deviation:

$$\frac{\forall t \in PG \wedge (\nexists o \in POD \rightarrow d \in t.D)}{ND}$$

where PG denotes process target set, $\rightarrow$ denotes matching.

For the second problem, generally speaking, there are two the criterions in deviation handling:

As per the target. When a deviation occurs, people, through discussing, regard that the initial targets are right, the software process should perform according to the initial targets.

As per the reality. When a deviation occurs, people, through discussing, agree that there are problems with the initial targets, the target should be adjusted to the reality.

The criterions of second problem actually are how the process support system makes the decisions when a deviation occurs. Because we discuss the deviation Under the PSEE tone, we think the weight of criterion A should be greater than B, and we generally adopt the criterion A.

For the third problem, how to solve the deviations according to different situations. In reference [14], there are five kinds of deviation handling policy: Abort, Inform user and abort, Ask user, Inform user and continue, and Continue. But we maintain that there are mainly three kinds of deviation handling policies:

1) Tolerating: Continue under the condition of without infringing on the target. (θ)
2) Adjusting the process model to adopt next execution of the process model: Process modelers find out that process outputs do not match the process target set after the execution of process, and they consider that there are problems in the process model, so the process model is needed to be adjusted. (ε)
3) Adding and improving this process execution: Process modelers find out that process outputs do not match the process target set after the execution of process, and they consider that the execution of process is unfinished, and there are some activities need to be added to finish the target. (ζ)

The characters after the above three kinds of deviation handling policies are notions of them. The handling way one

θ denotes that matching processing process outputs with the target set, modeler can find out all targets in the process outputs so that the process supporting system can adopt the tolerating handle way. The handling way two ε denotes that there are problems in the process model, so as to improve the process model. And the modelers are need to complete by this improvement process. The same to the handling way three ζ, it denotes that there are problems in the instantiating process. And in this process people are needed as well, so as to add and improve. To sum up, three ways are complement each other well, all of three ways need people to participate in, and all of three ways need to judge whether the process target set match process outputs by modelers. Only in this way, the whole judgment process can be finished.

B. An Example

A simple case of process deviation handling is given in this section. In order to present the handling schema, too complicated deviation sample are not included in this case, but we start from the simplest deviation to discuss. Firstly, we have an Enactment Process Model (EPM) M_1 as shown in Fig. 3:

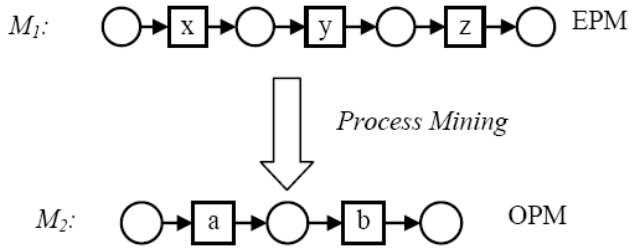

Fig. 3. Obtain the OPM through process mining.

Assuming that we have found the Observed Process Model (OPM) M_2, we can know that there is a deviation in this process enactment. However, more correlated conditions are needed to judge whether the deviation need to be rectified, so for the M_1 we have known that:

$$PG(M_1) = \{t_1, t_2, t_3\}$$

$$t_1 = (1, \{d_1, d_2\}, F, x)$$

$$t_2 = (2, \{d_3, d_4, d_5\}, F, y)$$

$$t_3 = (3, \{d_6, d_7, d_8\}, F, z)$$

$$POG(M_1)=\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9, o_{10}, o_{11}\}$$

And we have known the matching:

$$o_1 \rightarrow d_1, d_2$$
$$o_4 \rightarrow d_3$$
$$o_5 \rightarrow d_4, d_6$$
$$o_6 \rightarrow d_7$$
$$o_{10} \rightarrow d_8$$

Through the known conditions, we can find easily that there is not an output matching the description d_5 in target t_2. Consequently, we can judge whether we need to rectify or not according the standard of need to rectify deviation. We can divide into two cases to consider, based on the deviation handling criterion:

Case one: Assuming handling criterion A is adopted, we know that the initial target is right, so he software process should perform according to the initial targets.

Target is right, that is to say, initial EPM is correct, so the handle way we adopted is ζ, and we need add and adjust the M_2. Assuming that at this time:

$$PG(M_2)=\{t_1', t_2'\}$$

$$t_1' = (4, \{d_1, d_2, d_3\}, F, a)$$

$$t_2' = (5, \{d_4, d_6, d_7, d_8\}, F , b)$$

According M_1 and definitions, we can know that this deviation has actually resulted from the missing of activity '*y*', so we need to replenish *y*. But activity '*a*' has covered the activity '*x*', activity '*b*' has covered the activity 'z'. According to the above conditions, we find out that we only need to replenish activity '*c*' and the target of activity '*c*' $t_3' = (6, \{d_5\}, F, c)$. Then at this time the activity '*c*' is performed so as to achieve the process target set predefined.

Case two: Assuming handling criterion B is adopted, we know that there are problems with the initial targets, the target should be adjusted to the reality.

For the case two, there are problems in process target set PG, and the handle way we adopted is ε. Still assuming that:

$$PG(M_2)=\{t_1', t_2'\}$$

$$t_1' = (4, \{d_1, d_2, d_3\}, F, a)$$

$$t_2' = (5, \{d_4, d_6, d_7, d_8\}, F, b)$$

Because there is not an output matching the description d_5 in target t_2, we only need to delete the description d_5 in the EPM. In next instantiation of the process model, the situation, that process target set is not matching the process outputs, will not occur. Even though there is a deviation, we can also adopt the handle way θ.

IV. Case Study

In this section, we will use the approach proposed in this paper to solve the software process deviation, and give the relevant case studies. Through researching and analyzing the case, the thinking of whole paper and the solutions are further known about. This paper mainly refers the process algebra thinking and applies it on the research of process deviation. The process behavior space expression is put forward to construct the behavior space of software process. And through PBSE, we can detect the deviation efficiently, so that the deviation detected will be handled.

Firstly, the following conditions have been known:

1) The enacting process model as shown in Fig. 4 and this figure presents a software evolution process which comes from a company, where the meaning of activities is shown.
2) $PG(M_1)=\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$, where

$$t_1 = (1, \{d_1, d_2\}, F, a),$$

$$t_2 = (2, \{d_1, d_2\}, F, b),$$

$$t_3 = (3, \{d_3, d_4, d_5\}, F, c),$$

$$t_4 = (3,\{ d_6, d_7\},F,d),$$

$$t_5 = (5, \{d_3, d_8, d_9, d_{10}\}, F, e),$$

t_6 = (6, {d_{11}, d_{12}}, F, f),

t_7 = (7, {d_{13}, d_{14}}, F, g),

$t8$ = (8, {d_{15}, d_{16}}, F, h)

3) Target Set:

$POG(M_1)$={o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8 o_9, o_{10}, o_{11}, o_{12}, o_{13}, o_{14}, o_{15}} and the target matching rules:

$o_1 \rightarrow d_1,d_2$; $o_2 \rightarrow d_3$; $o_4 \rightarrow d_4,d_6$; $o_6 \rightarrow d_5,d_7$; $o_7 \rightarrow d_8$; $o_8 \rightarrow d_9$; $o_9 \rightarrow d_{10}$; $o_{10} \rightarrow d_{11}$; $o_{12} \rightarrow d_{12},d_{13},d_{14}$; $o_{14} \rightarrow d_{15},d_{16}$

The known conditions are a lot, and this mainly because the software process are very complicated process, which will involve more than one information.

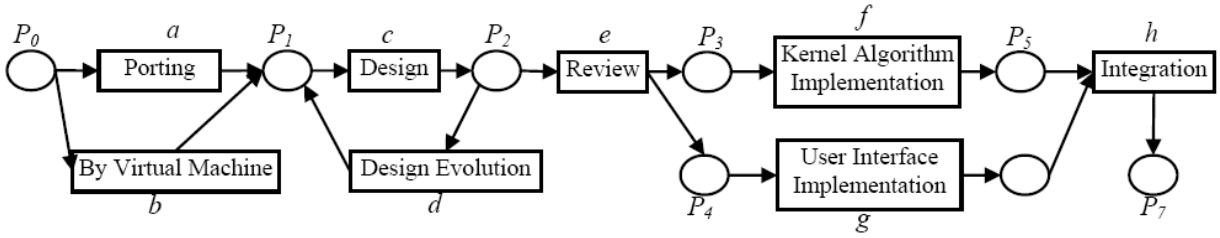

Fig. 4. Enacting process model M_1.

Step 1: We can get the PBSE corresponding EPM as follows:

We have had σ_1 = acdcefgh, σ_2 = bcesgfh

Aimed at σ_1 = acdcefgh, we have

$P_{[\#]} = (a+b).c!(d.c.)e.(f|g).h.\delta$
$P_{[\#a]} = c!(d.c.)e.(f|g).h.\delta$
$P_{[\#ac]} = !(d.c.)e.(f|g).h.\delta$
$P_{[\#acd]} = c!(d.c.)e.(f|g).h.\delta$
$P_{[\#acdc]} = !(d.c.)e.(f|g).h.\delta$
$P_{[\#acdce]} = (f|g).h.\delta$
$P_{[\#acdcefg]} = h.\delta$
$P_{[\#acdcefgh]} = \delta$

Aimed at σ_2 = bcegfs, we have

$P_{[\#]} = (a+b).c!(d.c.)e.(f|g).h.\delta$
$P_{[\#b]} = c!(d.c.)e.(f|g).h.\delta$
$P_{[\#bc]} = !(d.c.).e.(f|g).h.\delta$
$P_{[\#bce]} = (f|g).h.\delta$
$P_{[\#bcegf]} = h.\delta$

Through identifying the σ_2 in PBSE, we can find out that the activity '*s*' cannot be identified, and a deviation takes place. At the same time, there is not anything in the trackσ_2, so it is at the end of the track, that is to say, the activity '*s*' replaces the original activity '*h*'.

Step 2: When the deviation is occurred, the deviation should be considered whether need to handle or not. According to condition 2 and condition 3 we can know that activity '*h*' is replaced by the activity '*s*', and the descriptions of activity '*h*' are d15 and d16. So the missing of activity '*h*' will lead to that the rule $o_{14} \rightarrow d_{15}$, d_{16} cannot be matched. We need to use the standard of need to rectify deviation to judge whether need or not. Based on the standard of handling deviation, we can divided into two situations to consider.

Case one: Let us suppose that we adopt the deviation criterion A, so that the target set initially is right, and should execute according to the target. The target is right, that is to say, there is no problem in the original EPM, so we adopt the handling way ζ, which means that the OPM need to be added and adjusted. The deviation mainly is leaded by the replacement of activity '*h*' with activity '*s*'. Consequently, the activity '*s*' need to be defined as t_8'=(8,{d_{15}, d_{16}}, F, s), and this will conform to the process target set redefined.

Case two: Let us suppose that we adopt the deviation criterion B, so that there are some problems in the initial target and the target should be adjusted. For this situation, we will adopt the handling way ε. After people replacing the activity '*h*' with activity '*s*', whatever results obtained can be matched. And at next the instantiation of the process model, the situation that process target set cannot match the process outputs will not occur, even though there is the deviation, we can adopt the handling way θ. To sum up, the whole process of detecting and handling deviation have been discussed.

V. Conclusions

The problems on software process deviation receive more attention, because software processes are subject to permanent dynamic evolution. However, there still are some shortcomings of deviation research in the current software process domain, and the existing literatures cannot effectively solve the problem of software process deviation. Consequently, this paper put forward an approach to solve below problems based on the EPMM, referring the weak simulation thinking of process algebra, can effectively solve prevalent deviations problems and improve the quality of software products finally. In this paper, we firstly present a diagnosis architecture of process deviation based on EPMM. Secondly, in the aspect of software process deviation detected, we use the software process behavior expression to construct the behavior of process. Thirdly, in the aspect of software process handling, the deviation handle strategy is proposed to handle the deviation.

The future work mainly focus on 1) the method proposed in this paper will be consider how to be applied into the actual software process deviation, 2) a tool using the algorithm will be built, 3)whether we can try to establish a rule base to handle the deviation automatically without the participation of people.

References

[1] R. Balzer, "Tolerating inconsistency [software development]," presented at 13th International Conference on Software Engineering, IEEE, 1991.

[2] SEI, CMU, Capability Maturity Model Integration (CMMI) for Development, V1. 2. CMU/SEI-2006-TR-008, 2006.
[3] T. Li, *An Approach to Modelling Software Evolution Processes*, Berlin: Springer, 2008.
[4] L. Osterweil, "Software processes are software to," in *Proc. the 9th International Conference on Software Engineering*, IEEE Computer Society Press, 1987.
[5] C. Montangero, "The software process: Modelling and technology," *Software Process: Principles, Methodology, and Technology*, Heidelberg, Berlin: Springer, 1999, pp. 1-13.
[6] S. Arbaoui *et al.*, "A comparative review of process-centered software engineering environments," *Annals of Software Engineering*, vol. 14, no. 1-4, pp. 311-340, 2002.
[7] M. S. Li, Q. S. Yang, and J. Zhai, "Systematic review of software process modeling and analysis," *Journal of Software*, vol. 20, no. 3, pp. 524−545, 2009.
[8] K. Mohammed, R. Lbath, and B. Coulette, "A deviation-tolerant approach to software process evolution," in *Proc. Ninth International Workshop on Principles of Software Evolution: In Conjunction with the 6th ESEC/FSE Joint Meeting*, ACM, 2007, pp. 75-78.
[9] G. Cugola, "Tolerating deviations in process support systems via flexible enactment of process models," *IEEE Transactions on Software Engineering*, vol.24, no. 11, pp. 982-1001, 1998.
[10] K. Pohl *et al.*, "PRIME—toward process-integrated modeling environments: 1," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 8, no. 4, pp. 343-410, 1999.
[11] H. W. Jung and R. G. Dennis, "Evaluating the relationship between process improvement and schedule deviation in software maintenance," *Information and Software Technology*, vol. 5, no. 2, pp. 351-361, 2009.
[12] M. A. A. da Silva *et al.*, "Flexible Deviation Handling during software process enactment," in *Proc. 2011 15th IEEE International Conference on Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, IEEE, 2011, pp. 34-41.
[13] J. Lonchamp, "A Structured conceptual and terminological framework for software process engineering," in *Proc. International Conference on the Software Process Continuous Software Process Improvement*, 1993, pp. 41-53.
[14] G. Cugola, "Tolerating deviations in process support systems via flexible enactment of process models," *IEEE Transactions on Software Engineering*, vol. 24, no. 11, pp. 982-1001, 1998.

Rui Zhu was born in Henan, China in 1987. He received his bachelor of engineering degree in Software School from Yunnan University in 2010. He was selected to participate in a continuous academic project that involved postgraduate and doctoral study and received his master of science degree in Software School from Yunnan University in 2013. Currently, he is doing a doctorate in software engineering. He specializes in foundations of computer science, modeling, analysis, and design of software process using formal models such as petri nets and process algebra and related model checking techniques.