

# A Reuse-Oriented Process Component Representation Framework

Xiaohong Yang, Jing Lu, Ruzhi Xu, Guangfeng Pan, Jin Liu  
School of Computer Engineering, Shandong University of Finance  
Jinan, PA 250014, China  
{yangxh,lj,rzxu, pangf,liuj}@sdfi.edu.cn

## Abstract

*Process Reuse, as well as Product Reuse, can reduce the cost and the time to deliver software when compared to independently developed products. This paper proposes a systematic representation framework for the description and classification of Process Component (PC), which is composed of three levels of information description, namely PC's General Information Description, PC's Specification Description and PC's Data Description respectively. In order to facilitate the reuse and retrieval of PC, a facet-based PC classification scheme and an XML-based process modeling language (xPML) is presented. A reuse based process and project management system (P2MS) to support process improvement and control is also included in this paper.*

**Keywords:** Software process, Process component (PC), Process reuse, CMM

## 1 Introduction

Software Reuse is an important area in Software Engineering related to the reuse of the information produced during previous software development projects, in order to decrease the effort needed for a new project [1]. The main idea behind the reuse technology is to build software systems with higher quality and reliability factors in an economic and feasible process [2]. Thus, the productivity can be increased as the solution of previous projects is collected and retrieved to solve new problems.

Process reuse, as well as product reuse, can reduce the cost and the time needed to delivery software in a large-scale organization when compared to independently developed products [3]. The result of Rollenbach and Frakes's research shows that at least a tenfold improvement in time and effort to create a project process description occurs when we instantiate a reusable process instead of building the process from scratch [2,4].

Process reuse defines a wide area of research and practice related to different aspects of the reuse of knowledge obtained with previous successful projects. Many of the questions under this topic are related to the reuse of process assets. Other approaches [22] describe procedures to define generic process (or process templates), which are abstract models to be reused as the starting point for the modeling of processes in similar contexts. From a generic process, a user can adapt generic process descriptions to specific domains [6]. At CMM (Capability

Maturity Model) level3 or higher [16], the organization standard software process (OSSP) is repeatedly reused under different (but similar) contexts, to provides guidelines and support for organization's projects.

There are three prerequisites for process reuse: First, reusable objects are available; second, these objects must be useful; third, users must know how to reuse them. Therefore, the description and management of reusable process are the key for process reuse [22,24]. Process representation and classification becomes the major concerns for process reuse.

About process classification and description, Craig Rollenbach has presented a description scheme [4] of process (PC) classification and description, which is based on 3C (Concept, Content, Context) model [7], however, "3C" model is too abstract to implement in classifying and searching the PCs [8,9]. S.T. Fiorini [10] made a research on process classification and description based REBOOT model [11,12], but he didn't provide a detailed solution. Mr. D developed a set of process methods and symbols to define a single process in project STARS to explore the process model reuse [13].

However, process reuse not only satisfies the process defining description, but also provides relevant data for users to solve the problems[21,15]. Process data especially the knowledge acquired from the application domain and its implication for the resource allocation and scheduling of activities performed during the process execution must be recorded with the purpose to be reused for later process developments and control [3].

In this paper, a reuse-oriented process representation framework is proposed and a facet-based PC classification scheme and an XML-based process modeling language (xPML) are presented.

## 2 Basic concepts of process reuse

In this section, we firstly introduce some terminologies in the Process Reuse field. Following are the basic concepts of the Process, Process Reuse and Process Component.

**Definition 1:** *Process*, is a set of logically interacting activities to achieve a certain objective, which converts one or more inputs to output under recourses constrains and limited time.

Here, we formally define process as a ten-element-group has an explicitly defined

$$\text{Process} = \{\text{Ob, As, Ra, C}_1, \text{C}_2, \text{Ip, Op, Rs, Ag, Me}\}$$

Viz. {Objective, Activities, Relations between activities, Pre-Condition, Post-Condition, Input Product, Output Product, Resource, Agent, Metrics}

*Ob* (Objective) means every process has explicitly specified goals and the responsibilities.

*As* (Activities) is the core of a process, which converts input product to output product

*Ra* (Relations between activities) means the dependence relation between activities, it can be serial, parallel, iterative, collateral and nested, etc.

*Ip* (Input Product) and *Op* (Out Product), as compound products of atoms or other products, are the corresponding programs, documents and data input and output in the process or activities.

*Rs* (Resource) include human, material, technology, field and relevant information during the process.

$C_1$  (Pre-Condition) and  $C_2$  (Post-Condition) of the process are composed of activities restrict relationship between activities or products.

*Ag* (Agent) is an executor in a process or an activity. It can be a related person, program or a certain tool.

*Me* (Metrics) is used to measure and analyze the process, as well as to reflect process state, such as process schedule, cost and quality etc.

Process model and process instance represent two different abstract levels in process. Process model is the abstract definition and description for process structure and its attribute. It not only help the understanding and exchange in the software process, support communication in the software process, but also support the process analysis, instantiation management, measurement, process automation, process improvement and reuse. process instance is an executable process formed by the process model instantiation, which represents the specific elements of an actual process, such as detailed requirements and limitations such as schedule, resource, restrictions etc.

Software processes are software too [23]. Similar with developing a software system, a software process can be divided into several serial or parallel PCs. Process reuse and Process Component are defined as follows.

**Definition 2** *Process Reuse* is to build new process based on existing process Assets.

At CMM level3, Process Assets mainly archive the organization standard software processes (OSSP), which are developed, managed, and maintained by the software organization. They are composed of Software Life Cycle Model (SLCM), Organization Standard Software Process (OSSP), Process tailoring guides, Software Process Data, and Documents related to software process [16,18].

It is clear that process reuse includes process model reuse (such as SLCM, OSSP and other models, patterns and guides, etc.) and process information data reuse. The former one is based on well defined, relatively matured process model. The later one is to reuse the useful information acquired in the similar context or executing data available to evaluate current process, and create an executable process or project plan [14,20].

Obviously, it is difficult to reuse one software process in all projects in a large-scale software organization, which covers multi-application domains, However, some fine-grained process or sub-process and PC are much more reusable.

**Definition 3** *Process Component (PC)* is a process unit, which can be explicitly defined and identified. It encapsulates one or a series of activities to accomplish specified goals with recognized deliverable products.

PC differs in the granularity. According to PC definition, an independent software process can be treated as a PC, so can a sub-process of a software process. For example, sub-process such as “requirement analysis”, “spec documentation”, “coding”, “unit test”, and “integration test” are typically considered to be PCs. An activity or a task is considered to be a PC if it is strictly defined and meets all requirements in Definition 1 and 3. According to the definition above, a PC can be composed with other fine-grained sub-process components [22].

Reusable PCs are kept in the Process Component Library (PCL). To build a project’s defined software process: (1) we should firstly search for components in PCL. If the existing PC just same as expected one, we can directly use, (2) if there is no such PC, we can search for an approximate one from PCL, modify and instantiate it, and use it as a specialized PC, (3) If the existing PC is the same as the expected one, we can directly use it, (2) if there is no such PC, we can search for an approximate one from PCL. We can modify and instantiate it, and use it as a specialized PC, (3) if there is no similar PC, we can create a new PC which makes use of some fine-grained or low-level PCs by process assembling technology.

Such specialized PC and new assembled PC should also be stored in PCL. With the PCL accumulated, reusable PC provides strong and efficient support for process engineers and project developers to build process models and realize process dynamic adjustment, which brings great value to the process optimization and control.

### 3 PC Representation Framework

The efficient of the component query from PCL depends on the PC’s classification and description method. Following are So, we give a reuse-oriented PC representation framework.

PCs are Stored in the organization’s PCL, so when reusing a PC, querying Component in PCL is the first step. However, the efficiency of the component query depends on the process component classification and description method. A reuse-oriented PC representation framework is proposed in the following section.

#### 3.1 Reuse-oriented PC representation framework

According to the process reuse definition and analysis, process reuse includes the reuse of process related definition in process model, and the reuse of process information (knowledge and experience) [19,20]. So

reuse-oriented PC description content should include model definition of PC and PC related data information description. In order to make it convenient for ordinary user to search for PC without knowing the details of PC, general information of PC is a must in PC description.

As different users have different Objectives and need different contents for PC reuse, we present a universal reuse-oriented process representation framework. As depicted in Fig. 1, reuse-oriented process representation framework contains three levels of information descriptions, viz.: Process Component General information Description (PGD), Process Component Specification Description (PSD) and Process Component Data Description (PDD).

Different users will search for different information according to their Objectives. For example, a process modeling engineer wants to know the PC's detailed definition. He only needs to use PGD to search for the PSD instead of searching for the PDD. But for a project manager, PC's PDD may be more important, so he needs to use component's PGD to search for the PC's data.

Process component General information Description (PGD) is oriented to PC user and is applied in classification searching of PC. It primarily describes the Objectives and some specified information of PC, including functions, application domains, classifications, representation patterns and PC's contexts, etc. The description language used here is natural language or simple semi-formalized language familiar to common users.

PSD is oriented to process creators or namely process engineers. It is used to create, direct and support software process under running environment. It describes PC's activities and its relationships, process artifacts/products, agents, resources, pre-conditions and post-conditions etc., using non-formalized or semi-formal or even formal language as its description language.

Process component data description (PDD), is mainly oriented to process managers and process engineers. It is used to assess, manage, control and optimize the software process, which reflects the execution ability and data measurement when the PC is running. For a specific PC instance, it usually includes information about the start and finish time, efforts and its distribution, resource such as

number of team members, cost, process risk, process defect, product size and the context. PDD can be described by charts, figures, or other formats.

The descriptions above reflect a complete PC, covering the ten-element-group defined in Definition 1. Process objective (Ob) is described in PGD, and metrics (Me) in PDD. The specific definitions of process such as process activity and its relationships, input and output products, pre-condition and post-condition and other resource required in the process are represented in the PSD.

PC classification is based on the process general information description (PGD). When reuser wants to search for a certain PC specification, he can search for the PGD document. And if he needs some relevant data of a PC instance, he can also search for PGD, finding the required PC and then access the corresponding PC's data.

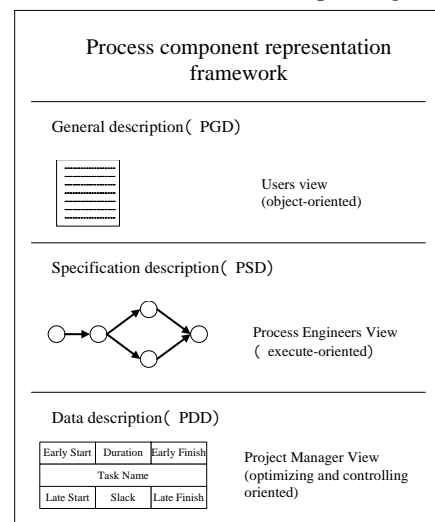


Fig. 1 Reuse-oriented PC Representation Framework

Process component general information description (PGD) depends on PC's classification schema. The quality of PGD may affect the searching efficiency of PC. PC's specification (PSD) is of great importance for process engineer to create new process by using such PC; meanwhile PDD can help process manager, especially project manager, to make proper project scheme and right process control decisions. Next is the discussion about the above and the corresponding solutions.

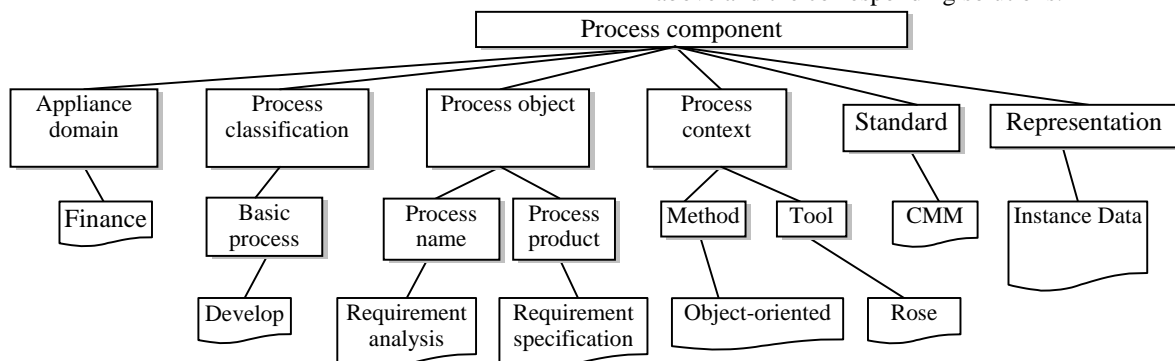


Fig. 2 PC's facet classification scheme

PGD describes PC's objective, and its basic information, such as process name, process version, process author and user's information, application domain, classification and representation, etc. PGD is oriented to PC user, which is mainly used in classification of the PC.

This paper describes the PC's general information based on Faceted Classification. Every component is depicted by several facets by a group of terms, namely description key words respectively. By endowing each set of facets with relevant terms, users can search in component library using the specific terms of a set of facets and find out the required component. Meanwhile, in order to solve the inconsistency between the term value given by users and its correspondence in the component library, users use the software tool to analyze thesaurus and then enhance the completeness and preciseness of the components. Facet classification pattern is agile and easy to add new facet to it, which makes the pattern more flexible.

As process varies in different patterns, it can hardly design a universal PC description scheme. Next we put forward the description policies based on the facet classification:

- Facet classification design is mainly for PC search and reuse. Facet description content should not involve PC's activity details, but should consider mining historical information for PC reuse.
- The content described on each facet should be vertical and complete. The content of each facet should be comparatively independent from each other, and able to present reuse general information.
- The facet number on one level should be proper. 2~7 facets are suitable for a level.
- The depth of the hierarchy should be rational, 2~5 levels are fine.

Fig. 2 presents an example of a classification plan based on facet, which contains 6 facets, namely application domain, process classification, process Objective, process context, standard and representation. The description of

information and document type on each facet and sub-facet is particularly defined in Reference [17].

Each software organization itself can define its process facet classification description scheme based on the facet classification direction principles given in this paper, referring to PC facet classification description document type definition (DTD), and according to specific process characteristics and requirements.

### 3.2 PC's Specification Description(PSD)

Process component specification is combined with 2 parts, one is description of PC structure, depicted by process component activity flow chart; the other is description of process attribute, described in the PC attribute table.

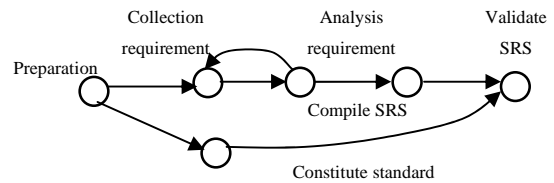


Fig. 3 PC's Activity Flowchart

Activity flow chart describes the serial and parallel relationships between activities in the process component. Fig. 3 is an activity flow chart of a requirement of a PC, which describes a demand analysis process composed of different activities, such as preparing period, collecting requirement, analyzing requirement, making requirement evaluation standard, compiling and validating software requirement specification (SRS). The arrows in the figure 3 present the relationships between these activities.

Process attribute table describes the attributes of PC, such as activities, activity agents, inputs and outputs products, pre-conditions, post-conditions and other resource attributes, etc. Table 1 is an attribute table of one requirement analysis process.

Process name	Activity description	Agent/role	Pre condition	Input product	Post condition	Output product	Resource	Metrics items
Requirement analysis	Preparing Collect requirement Analyze requirement & compile SRS Constitute standard Validate SRS	A	Project start	User requirement	Pass SRS validating	SRS	Rational-Rose, etc.	Workload Time Defect num. SRS size Risk

Table 1 PC's attribute table

Each PC has one PSD, so the structure, attributes and their interaction relations of the PC are all very explicit. Therefore, every PC can be reused many times under the same context by process engineers or project developers. Moreover, fine-granularity sub-process component, which has been defined completely and explicitly, can develop a larger PC and finally form a complete software process.

### 3.3 PC's Data Description(PDD)

The data of a PC is collected during its execution. According to CMM, the measurement data of a PC is not only related to the environment during the running time of process component, but also related to organization's process capability maturity. So it can only be useful when both process component measurement data and relevant environment of process are recorded. And at the same time, the capability baseline can be calculated by statistically analyzing the data of the same process component in different projects so as to predict the possible result of this process in the future.

In PGD, the context and environment of the process component are described, following we will analyze what data or information should be encapsulated in the process component.

For each project, its process statement is presented by parameters related to time, cost and quality, which compose the "Project Triangle". Accordingly, a process component metrics items should also reflect time, cost and quality. As the degree of these 3 metrics items depend on the size of the product and risk management is an important part of project management, the risk data related to these 3 factors also need to be measured. Process

Component metrics (ProComMes) can be simply described as follows:

ProComMes = Me {Size, Cost, Time, Quality, Risk}

Size is measured by function point, lines of code number and page number of product;

Cost is the human resource that software project actually spent, which takes efforts as the metrics items.

Efforts are represented by person-hour, person-day, person-month, etc.

Time reflects the starting time and finishing time of the process executing, or its lasting time.

Quality may contain a lot of metrics items. Here we take defect numbers and the severity of process component products as the metrics items.

Risk is the possibility of unexpected results of the above basic factors or deviation from the plan and its loss scope as metrics items.

Otherwise, the measure results above should be compared with the planned data, and then we need to record the reason for the deviation and make the analysis. From these data, we can obtain more valuable information.

## 4 Process Description Language

Process modeling language is the formal description tool of process modeling. In this paper, we adopt XML as the metal language of process and process component description.

XML is a widely used Extensible Markup Language. It employs abundant lingual description ability. The expansibility of XML can provide mini-adjustment lingual description ability for specific component library. A component description system combined with public component description template and user-defined component description template can be imagined, which makes the agility, abundance and moderate preciseness of the component description to a higher level.

Moreover, the document type definition ( DTD ) function of XML endue a level intension toward process modeling and process component description , which matches the process structure. This paper, based on chrematistics of XML, presents xPML ( Process Modeling Language Based on XML ) , a kind of definite description process modeling and process component metal language. Fig. 4 is the basic hierarchy structure chart of xPML.

From Fig. 4 we can see that xPML can provide a complete description support for the above 3 information levels in "reuse oriented process component representation framework".

For general information and data description about process component, xPML maps facet classification scheme and metrics items to "element/parameter-value" structure of xPML. But for process specification description, formalized language description is generally used, so we need to translate specification description to xPML DTD according to formalized language BNF paradigm.

According to the definition of xPML DTD [17], we give an XML description document of PGD as follows. Its process classification is development, function is requirement analysis, appliance domain is finance, and it applies to CMM standard.

```
<process component>
  <process object>
    <process name>requirement analysis</process name>
    <process product>requirement specification
  </process product>
  </process object>
  <application domain>finance</application domain>
  <process class>
    <basic process>develop</basic process>
  </process class>
  <standard>CMM</standard >
  <representation>instance plot</representation>
  <process context>
    <method>object-oriented</method>
    <tool> Rose </tool>
  </process context>
</process component>
```

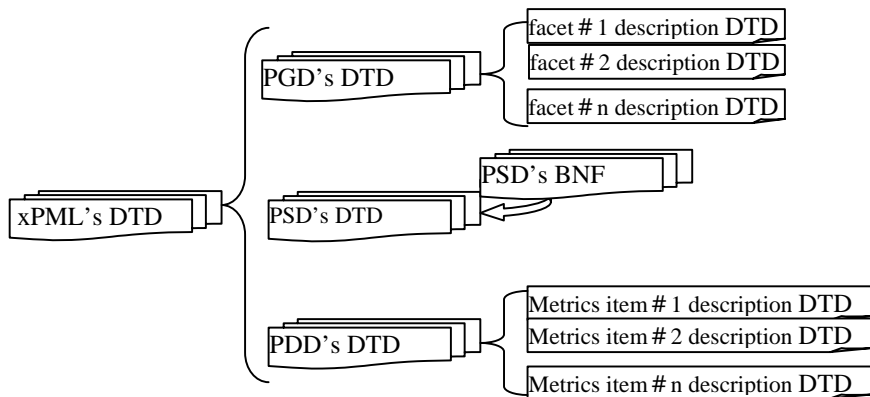


Fig 4 DTD structure of PML

xPML is not only tailorable, extensible and structural, but also allows organization and person to build proper remark set according to one's needs. Therefore the depth and width of description data can be expanded under maintaining preciseness of data structure. This characteristic meets the requirement of component classification mechanism, makes component description system continually attract new classification mechanism or add new classification standard in each classification mechanism, and has solved the class limitation problem brought by continually adding new components.

## 5 Reuse-based P2MS

Based on process component representation framework proposed in this paper, we develop a process management and project management system (P2MS), which supports the process management and control. As depicted in Fig.5, P2MS supports the software process definition, instantiation, analysis, execution, monitor and dynamic adjustment.

P2MS consists of three sub-systems according to user classification: (1) process component management and organization system, (2) process component reuse and feedback system and (3) process execution and measure system.

Process component management and organization system is oriented to process engineers, and it can be divided into 3 supporting systems: reusable process component editor, optimizer and manager. Editor is responsible for distilling, describing and inputting reusable process component. Optimizer can validate and optimize the reusable process component based on case analysis, static analysis and simulation technology. Manager is responsible for the feedbacks and maintenance of reusable process component library.

Project process management system is mainly oriented to project manager, and it can be divided into 3 supporting sub-systems: process instantiation, process optimization, process adjustment. Process sub-system is based on process tailoring and instantiation. A project manager can

cut the organization defined standard software process (one of the standard software process component in the process component library), according to the characteristics of a project, and build a project-defined software process, or he can define a project process based on several smaller-granularity process components in the library.

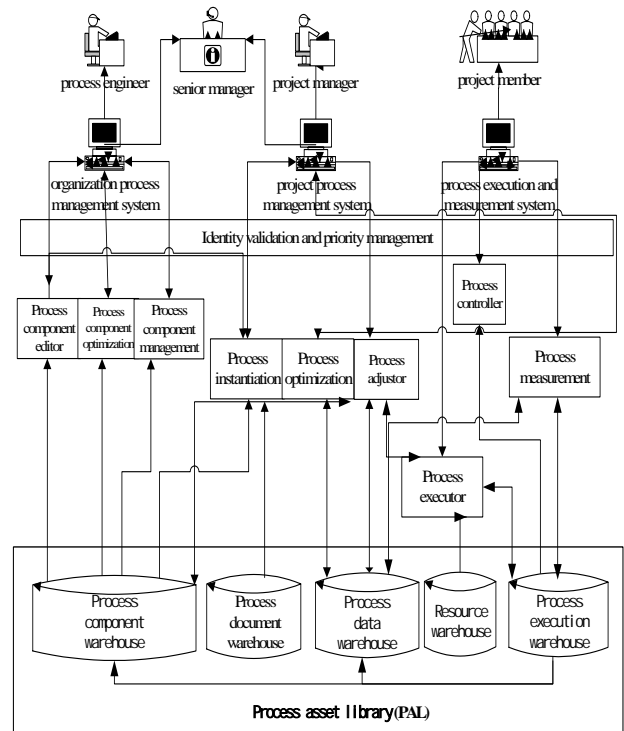


Fig.5 Architecture of P2MS

Process optimization subsystems use historical data in the process data library to optimize the project process through static analysis and dynamic simulation technology. Process adjustment/feedback sub-system finally accomplishes the project plan based on the results from process analysis and optimization, and provides feedback

the reuse information of this project process to reusable process component library and process data library.

Process execution and measurement system, is mainly oriented to project team member. It is combined with 2 sub-systems: process measurement and process execution/control. Process measurement sub-system is responsible for measuring and analyzing project development. Process executing/controlling sub-system helps project manager and project member to supervise the project executing and track latent risks through several process views.

## 6 Conclusions

In this paper we proposed a systematic representation framework to describe Processes Components (PCs), which is composed of 3 levels of information of PGD, PSD and PDD respectively. To facilitate the reuse and retrieval of process component, a facet-based PC classification scheme and an XML-based process modeling language (xPML) is presented. Finally, A reuse-based process and project management system (P2MS) to support process improvement and control is introduced in this paper.

## Reference

- [1] Isoda, S. Experience Report on Software Reuse Project: Its Structure, Activities and Statistical Results, In: International Conference on Software Engineering, 14., 1992, Melbourne. Proceedings. ACM Press, 1992.
- [2] Frakes, W. Systematic Software Reuse: a Paradigm Shift. In: International Conference on Software Reuse, 3., 1994, Rio de Janeiro. Proceedings... Los Alamitos, California: IEEE Computer Society Press, 1994.
- [3] Reis, R.; Reis, C.; APSEE: A knowledge-based software process enactment and modeling infrastructure. Technical Report. Porto Alegre: PPGC-UFRGS, 2000.
- [4] Rollenbach, C., Frakes, W. Software Process Reuse in an Industrial Setting, Fourth International Conference on Software Reuse, Orlando, Florida, IEEE Computer Society Press, Los Alamitos, CA, 1996.
- [5] Amber, S.W. Process Patterns: Building Large-scale systems using Object Technology, Cambridge Univ Press 1998.
- [6] Lonchamp, J. A Structured Conceptual and Terminological Framework for the Software Process Engineering. In: International Conference on The Software Process, Berlin. Proceedings... Berlin: IEEE Computer Society Press, Mar. 1993.
- [7] Tracz, W., Where Does Reuse Start?. Proc. Reality of reuse workshop, Syracuse University Case Center, January 1990.
- [8] L.Latour, T.Wheeler, and B.Frakes, Descriptive and prescriptive aspects of the 3Cs model, In Proceedings of the Third Annual Workshop, Methods and tools for Reuse. CASE Center Technical Report number 9014, Syracuse University, January 1990.
- [9] Jean-Marc MOREL & Jean FAGET. The REBOOT Environment” BULL S.A. Rue Jean JAURES, F-78340 LES-CLAYES-SOUS-BOIS, FRANCE.
- [10] Fiorini, S.T., Leite, J. C. S. P. Process Reuse Architecture, PhD Thesis, Department of Information, Pontificia University, 2001.
- [11] Rubén Prieto-Diaz and Peter Freeman. Classifying software for reusability. IEEE Software, Vol. 4, pp. 6-16, Jan. 1987.
- [12] O’Connell, F.: How to Run Successful Project -The Silver Bullet, Prentice Hall Europe, 1996.
- [13] Radice, R., Roth, N., O’Hara, Jr., A., Ciarfella, W. Programming Process Architecture. IBM Systems Journal, Vol.24(2), 1985
- [14] Xu Ru-zhi, Qian Le-qiu. CMM-based Software Risk Control Optimization. Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration, Las Vegas, 2003.499-503.
- [15] Lima, C. A Software Process Manager for the PROSOFT Environment. CPGCC-UFRGS. 1998. M.Sc. Thesis. (<http://www.inf.ufrgs.br/~prosoft>)
- [16] Mark C. Paulk, Charles V. Weber, etc. Key Practices of the Capability Maturity Model, SM, Version 1.1. Technical Report, CMU/SEI-93-TR-025, ESC-TR-93-178, 1993
- [17] Xu Ruzhi. Reuse-based software process improvement and control optimization. PHD Dissertation, Fudan University, 2004.
- [18] Pankaj Jalote. CMM in Practice: Process for Executing Software Projects at Infosys. Addison-Wesley, 2000.
- [19] Zhou Zhiying. Reusability Based on P-F Method for Software Process Modeling. Journal of Software.2001,12(8):1258-1264.
- [20] Xu Ruzhi, Qian Leqiu, Zhao Wenyun. Research on Metrics-based Software Process Control Optimization. Journal of Electronics, 2003, 31(12A):1206-1209.
- [21] Mark C. Paulk, Charles V. Weber, etc. Key Practices of the Capability Maturity Model Ver1.1. Technical Report, CMU/SEI-93-TR-025, ESC-TR-93-178, 1993.
- [22] Kellner, M. Reuse of Process Elements. In: Boehm, B.; Kellner, M.; Perry, D. (Ed.) International Software Process Workshop. IEEE Computer Society, 1998. p. 19-23.
- [23] L.Osterweil, “Software Processes are software too”, In Proceeding of International Conference On Software Engineering, 1987, 2-13.
- [24] Xu Ruzhi, Qian Leqiu. etc.. Research on matching algorithm for XML-based software component query. *Journal of Software*. 2003,14(7):1195-1202.