ISSN: 1945-3116(Print), 1945-3124(Online) Volume 2, Number 3, October 2009

Journal of Software Engineering and Applications

Chief Editor : Dr. Ruben Prieto-Diaz

www.scirp.org/journal/jsea

Scientific Research

Journal Editorial Board

http://www.scirp.org/journal/jsea

J. Software Engineering & Applications, **2009, 2**
Published Online October 2009 in SciRes(www.SciRP.org/journal/jsea)

CONTENTS

Volume 2 Number 3 **October 2009**

Journal of Software Engineering and Applications (JSEA)

Journal Information

SUBSCRIPTIONS

The *Journal of Software Engineering and Applications* (Online at Scientific Research Publishing, www.SciRP.org) is published monthly by Scientific Research Publishing, Inc., USA.

E-mail: jsea@scirp.org

Subscription rates: Volume 2 2009

Print: $50 per copy.

Electronic: free, available on www.SciRP.org.

To subscribe, please contact Journals Subscriptions Department, E-mail: jsea@scirp.org

Sample copies: If you are interested in subscribing, you may obtain a free sample copy by contacting Scientific Research Publishing, Inc. at the above address.

SERVICES

Advertisements

Advertisement Sales Department, E-mail: jsea@scirp.org

Reprints (minimum quantity 100 copies)

Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.

E-mail: jsea@scirp.org

COPYRIGHT

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:

E-mail: jsea@scirp.org

J. Software Engineering & Applications, 2009, 2: 137-143
doi:10.4236/jsea.2009.23020 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship

Graylin JAY[1], Joanne E. HALE[2], Randy K. SMITH[1], David HALE[2], Nicholas A. KRAFT[1], Charles WARD[1]

[1]Department of Computer Science, University of Alabama, Tuscaloosa, USA; [2] Department of Management Information Systems, University of Alabama, Tuscaloosa, USA.
Email: {tjay, rsmith, nkraft, cward}@cs.ua.edu, {jhale, dhale}@cba.ua.edu

Received April 21st, 2009; revised June 9th, 2009; accepted June 12nd, 2009.

ABSTRACT

Researchers have often commented on the high correlation between McCabe's Cyclomatic Complexity (CC) and lines of code (LOC). Many have believed this correlation high enough to justify adjusting CC by LOC or even substituting LOC for CC. However, from an empirical standpoint the relationship of CC to LOC is still an open one. We undertake the largest statistical study of this relationship to date. Employing modern regression techniques, we find the linearity of this relationship has been severely underestimated, so much so that CC can be said to have absolutely no explanatory power of its own. This research presents evidence that LOC and CC have a stable practically perfect linear relationship that holds across programmers, languages, code paradigms (procedural versus object-oriented), and software processes. Linear models are developed relating LOC and CC. These models are verified against over 1.2 million randomly selected source files from the SourceForge code repository. These files represent software projects from three target languages (C, C++, and Java) and a variety of programmer experience levels, software architectures, and development methodologies. The models developed are found to successfully predict roughly 90% of CC's variance by LOC alone. This suggest not only that the linear relationship between LOC and CC is stable, but the aspects of code complexity that CC measures, such as the size of the test case space, grow linearly with source code size across languages and programming paradigms.

Keywords*: *Software Complexity, Software Metrics, Cyclomatic Complexity

1. Introduction

Software complexity is traditionally a direct indicator of software quality and cost [1–6]. The greater the complexity (by some measure) the more fault prone the software resulting in higher cost. Much effort has gone into identifying techniques and metrics to 'measure' the complexity of software and software modules [7]. Logically, many of these measures have been shown to be correlated in some manner. Understanding these relationships is important to understanding and evaluating the metrics themselves and ultimately in reducing software development and maintenance efforts. This research reexamines the relationship between Lines of Code (LOC) and McCabe's Cyclomatic Complexity (CC) a traditional complexity metric.

First introduced in 1976 [8], McCabe's Cyclomatic Complexity (CC) is intended to measure software complexity by examining the software program's flow graph. In practice, CC amounts to a count of the "decision points" present in the software. CC can be calculated as:

$$CC = E - N + 2P$$

where

E is the number of edges,

N is the number of nodes, and

P is the number of discrete connected components.

CC was originally meant as a measure of the size of the test case space [8].

While numerous studies [1–3,9] have examine the relationship between LOC and CC, few have made it their central point of inquiry. As a result, while many state, sometimes strongly, that LOC and CC have a linear relationship, few investigate statistical issues such as the distribution of variance among LOC and CC. Shepperd, for example, uses data from previous studies to argues that CC was often "merely a proxy for ... lines of code" [9]. Many investigators either consciously or serendipitously avoid the issue entirely by using mixed metrics

such as error density or adjustments for size [5, 10]. Others investigating the relationship of CC to some other factor explicitly tested for a detrimental multi-collinear effect from LOC [11]. While previous studies have indicated the large role that LOC seems to play in CC [12], they stop short of claiming a general model of the relationship. While we do not seek to settle the issue, it is for these reasons that this research reexamines the relationship of LOC and CC in the context of a large empirical study.

2. Study Methodology

As a baseline and to confirm the LOC/CC relationship results reported in the literature, a pilot study looked at 5 NASA projects from the PROMISE Software Engineering Repository [13]. The PROMISE Repository is a collection of publicly available datasets for software engineering researchers. The NASA projects were originally archived in the NASA Metrics Data Program. Table 1 shows the Pearson Moment of Correlation between LOC and CC.

The correlation is remarkably high (average 0.896), yet does have a significant variance. When expanding on this pilot, variance was examined closely for the larger sample population.

2.1 Sample Population

For the larger study, the SourceForge.net (SourceForge) software repository was chosen because of its breadth and popularity [14]. SourceForge is the most popular public software repository on the Internet and is second only to Download.com as the most popular provider of software on the web [15]. SourceForge is home to projects actively sponsored and developed by companies such as HP [16] and IBM [17] as well as academic and other open-source projects. SourceForge is home to over 170 thousand different software projects all with their full codebases publicly available.

2.2 Population Candidate Stratification

Based on the observations of the PROMISE Repository, the large sample population of SourceForge projects was stratified based on three popular languages: C, C++ and Java. Identification of the implementation language is part of project creation on SourceForge. This self-reported information was used to establish three subject candidate populations. Table 2 shows the number, by language, of candidate projects considered for this study as well as the number of projects actually selected and analyzed.

Projects that mixed candidate languages were eliminated. That is: while a project that employed Python and C was considered an acceptable candidate, a project that used Java and C++ was not.

2.3 Subject Selection

One thousand subjects were randomly chosen from the stratified lists (column two of table 2). All of the chosen subjects needed to employ the Subversion (SVN) version control system [18] rather than the more traditional CVS, so this criterion was used to further discriminate amongst projects. The speed and reliability of SVN made this experiment practical. The choice of SVN over CVS did not affect the sample statistics. A complete discussion of this issue and other analysis is given in the Results section.

Table 1. Representative NASA projects and their pearson moment between LOC and CC

Project	*Language*	*Pearson Moment*
spacecraft instrument	C	0.94
real-time predictive simulation	C	0.82
data storage manager	C++	0.90
science data processor	C++	0.96
satellite flight software	C	0.86

Table 2. Candidate population sizes (in projects) and final number of active subjects

Language	*Candidate Projects*	*Selected Active Projects (at least one source file)*
Java	21,739	728
C	13,336	749
C++	15,194	747

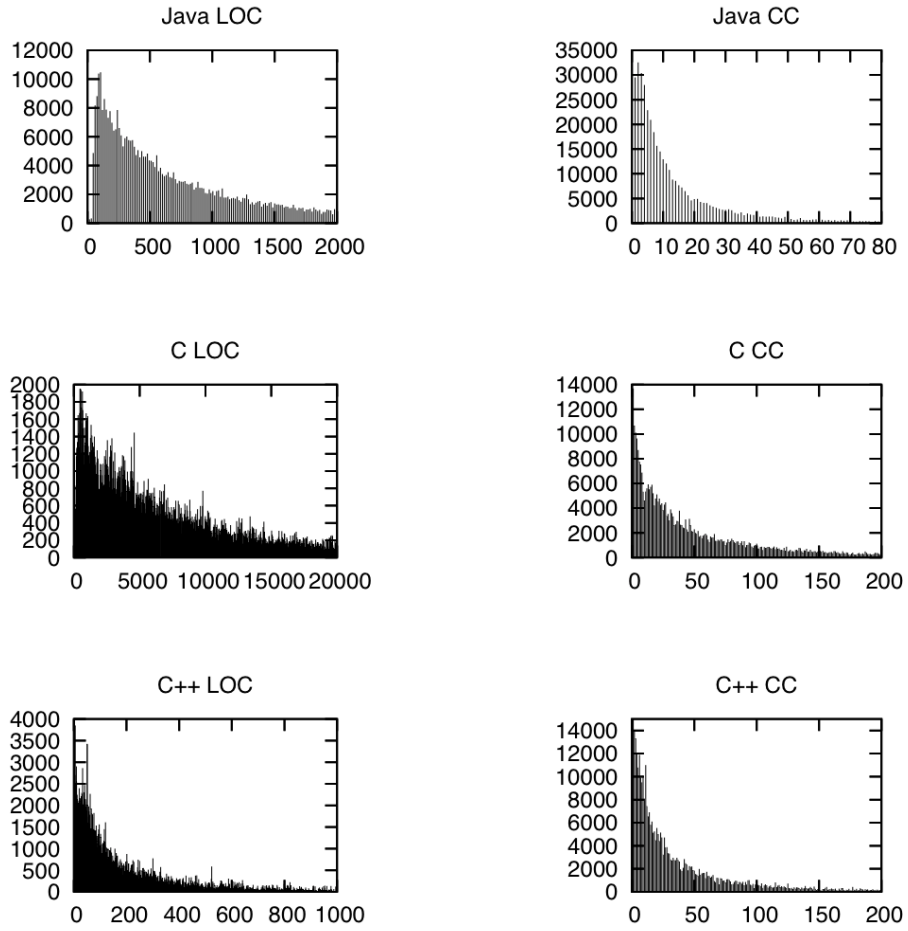


Figure 1. LOC and CC versus number of source files

Given its breadth and scope, many of the projects in SourceForge are no longer active or are simply non-existent – project space exists but no files exist. Rather than risk tainting the samples by overstating the "active" projects, the one thousand subjects for each language were randomly selected from the candidate populations with no regard to project activity level or completeness. This resulted in a number of the projects in the sample not having any source files at all. The final number of "active" (one or more source files) projects contained in the final samples is given in Table 2 (column three). It is noted that the ratio of active to non-existent projects seems fairly constant between languages (3% - 5%). When the selection process was finished, the sample projects to be analyzed (about 750 per language) consisted of more than a quarter terabyte of source code.

2.4 The Metric Tools

To collect the actual CC and LOC metrics, the study employed two tools. The main tool was the popular commercial tool RSM (Resource Standard Metrics). RSM was chosen because of its ISO certification and its use at various Fortune 100 companies [19]. For comparison, the C and C++ Code Counter (CCCC) [20] open-source tool was employed. CCCC and RSM provided similar results for LOC and CC.

3. Descriptive Statistics

The study examined roughly 1.2 million files, over 400,000 C files alone. Figure 1 shows the distribution of LOC and CC for each language.

Before proceeding with any regressive or other correlation analysis the assumption of normality was confirmed by an Anderson-Darling analysis. At a 95% confidence level, it was concluded that all distributions were log normal distributions, save for C language files. The C language samples' LOC and CC instead have a Pareto (also known as a Bradford) distribution. The Pareto distribution is very similar to the log normal distribution except that its population distribution is less even. In this case, relatively fewer projects account for more of the CC and LOC. Since both log normal and Pareto have similar curvature issues, the rest of our analysis were performed in a log adjusted space. An example of such

an adjustment is presented below in Figure 2, which shows the log adjusted LOC distribution for the C++ samples. These adjustments result in almost ideal normal curves for the sample populations.

3.1 Variance Issues

To test the assumption of evenly distributed variances, A Breusch-Pagan [21] test was performed on each of the samples with a significance level of .05. In each case homoscedasticity was rejected. This indicates that the variance within the sample populations was not uniform. This is a significant finding. Equality of variance is a required assumption for most traditional forms of regression. These traditional forms of regression are exactly the types of regression used in previous research. Our results indicate that this unevenness is more than just a theoretical concern. Below it is shown that a Pearson analysis is skewed compared to a more robust analysis.

4. Results

The Pearson Moment was calculated between the log of the LOC and the log of the CC for the samples as was the explanatory power of the log of the LOC over the variance of the log of the CC. These log transformations adjust for the curvature present in the samples' log normal distributions. Table 3 gives the Pearson Moment and variance by language and tool. The CCCC tool could not process Java files.

Earlier, it was discussed that samples were limited to those that utilized SVN. As a check that this did not invalidate the results, a small random sample of 32 projects per language were selected that utilized another open-source versioning system (CVS). Table 4 gives those results.

Table 3 and Table 4 below indicate a strong linear correlation between the log of LOC and the log of CC, and hence between LOC and CC. This correlation is strong regardless of language. When CCCC failed to be capable of processing a source file in a project, the project was removed from the CCCC sample. Despite the fact that this meant CCCC's sample was differentiated, the two tools still both indicate the same strong correlation.

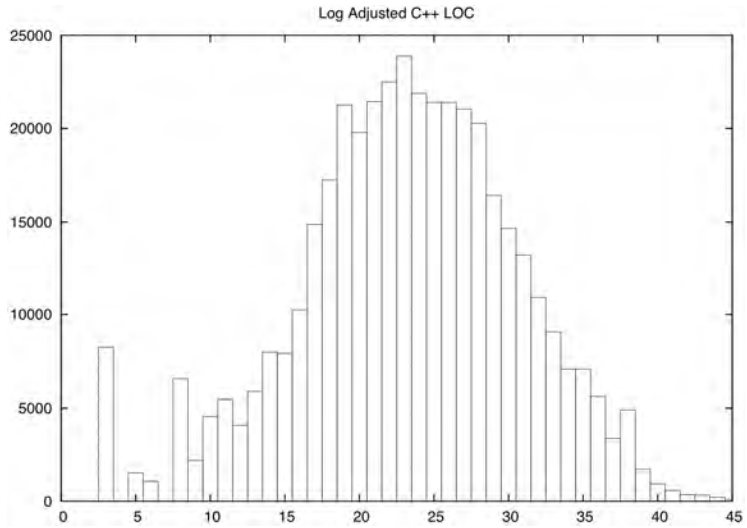


Figure 2. Log adjusted C++ LOC distribution

Table 3. Pearson moment in log adjusted space by language and tool

Language	*Tool*	*Files*	*Pearson Moment*	*Percent of Variance*
Java	RSM	480,336	0.88	78.3
	CCCC	NA	NA	NA
C	RSM	401,474	0.88	78.4
	CCCC	399,483	0.91	82.7
C++	RSM	411,718	0.87	76.2
	CCCC	410,051	0.85	72.9

Table 4. Pearson moment in log adjusted space by language (32 CVS projects each)

Language	*Pearson Moment*	*Percent of Variance*
Java	0.91	82.5
C	0.87	76.8
C++	0.93	86.4

Table 5. Coefficient of determination for Siegal repeated median regression and "equivalent" pearson moment

Language	*Coefficient of Determination*	*Equivalent Pearson Moment*
Java	0.87	0.93
C	0.93	0.97
C++	0.97	0.98

Concerns over variance made it necessary to run a more robust test than Pearson. The test chosen was the Siegal repeated median regression, a technique known to be robust against heteroscedasticity and tolerant to up to 50% of the data points being outliers [22]. Siegal is computationally intensive. To accommodate the computational complexity given the sample size, 3000 data points were randomly sub-selected from the samples. A linear model for a each sub-sample was created using repeated median regression. These models were then used to predict CC for *all* the samples of a language population based solely on LOC. To assess how predictive these models were, their coefficient of determination were computed (see column two of Table 5.). So that the accuracy of our repeated median regression models could be compared to more traditional models, the equivalent Pearson Moment for each coefficient were also calculated. These are what the Pearson Moments in a traditional model would have to have been in order to account for the same amount of variance as our Siegal-based models. All of the calculations here described were performed in the same log adjusted space as with our previous Pearson Moment calculations. The results for each language are shown in Table 5.

As shown in Table 5, once the study accounted for issues of variance LOC and CC, extremely accurate linear models were developed. It is worth reiterating: while the models were developed using sub-samples, the values in Table 5 are from applying the model to the *whole* populations. Our models can use log of LOC to explain all but 13% of the log of CC's variance (on average they explain 90% of the variance). Based on these results we propose:

LOC and CC are measuring the same property. Whether this means that LOC and CC are merely estimates of each other or if they are both estimates of some third factor is left as an open question. Regardless, the fact that LOC and CC do measure each other indicates that models using one or the other must be careful of collinear effects.

Figure 3 shows how similar the models are for each language. Figure 3 shows the graph of the Siegal repeated median model for each language. For clarity's sake this graph is in the un-adjusted space.

4.1 Model Validation

It is worth reiterating how our Siegal repeated median models were developed. They were built using data from a small portion of each language population and then used to predict attributes of the entire, larger, language population. This is an important point because it means that the link between LOC and CC that the models represent have been externally validated as indicated by Zuse [23]. We have used LOC to accurately predict CC in a large (hundred of projects, thousands of files) varied (professional, amateur, and academic) population. SourceForge provides a heterogeneous cross-section of the general software population.

5. Threats to Validity

It would be misleading to think that this study concerning metric directly mitigates internal validity threats. While they are considered metrics in their own right, there is a great deal of dispute as to how to practically "measure" CC and LOC. We attempt to address this issue through our use of multiple measures in the form of our two toolsets. However, this is by no means an exhaustive solution to the problem and was not possible for Java.

We present strong statistical evidence for the general applicability of our findings across languages, paradigms, and skill-sets. We stress that while this generally applicability is statistically true, it is only true in aggregate. The general applicability to any given project is still an open issue.

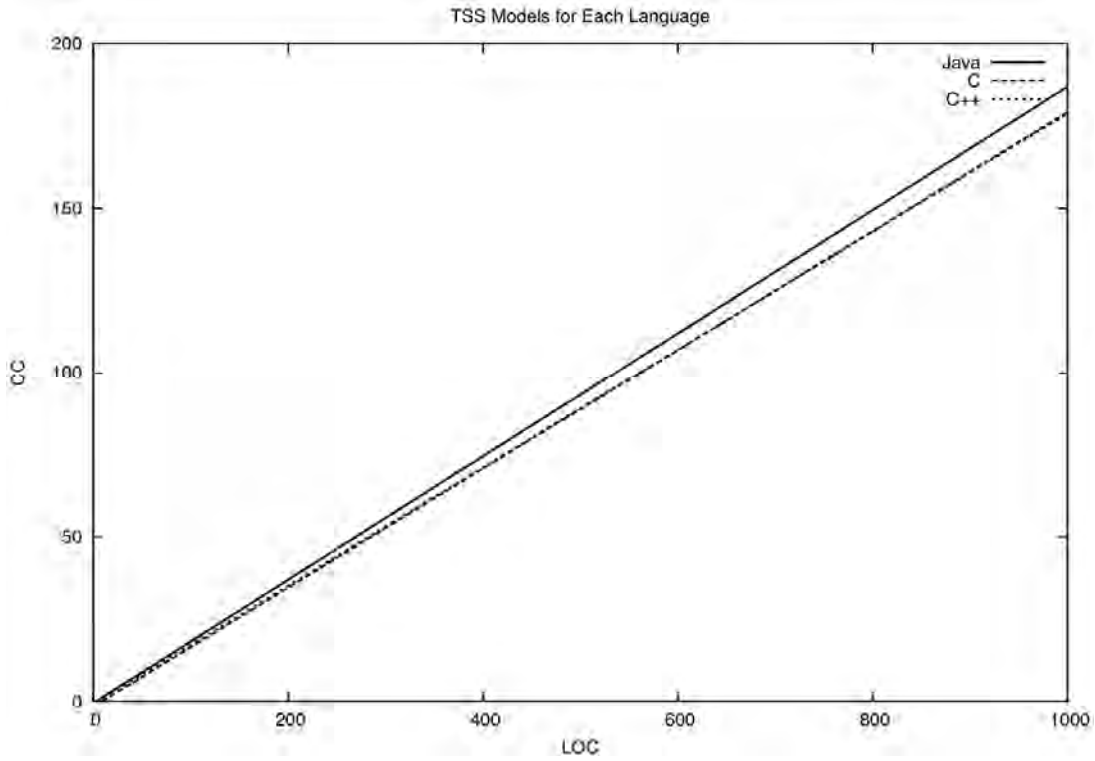


Figure 3. Siegal repeated median model for each language

6. Discussion

It is known that accurately estimating collinear factor's linearity can be difficult. By utilizing the large sample size in this study, the co-linearity of CC and LOC was statistically determined. These results help to address some of the contradictory findings in previous studies [2,3,6,9,24,25,26] regarding CC, LOC, errors, maintenance effort, and so forth. Factors as linearly related as LOC and CC should be considered collinear. Models that fail to properly bind together collinear or multi-collinear factors will often have unstable explanatory power. The instability of predictions based on collinear factors can provide a theoretical explanation for so many contradictory findings. While likely not the dominant factor, this effect could also provide a partial explanation for why researcher such as Menzies et al. have discovered so much more predictive power in hybrid predictors than so-called mono-metrics [27]. In support of Menzies et al, hybrid metrics can properly bind these factors where mono-metrics cannot. Mono-metrics lack needed information that is captured by combined or hybrid metrics.

The linear relationship between LOC and CC raises has several direct implications for software maintainers and evolution management.

CC has no (or very little) explanatory power of its own. This implies that indicators that rely on CC may more easily be calculated and normalized by using LOC. Calculation of CC requires some cost however small. The results from this study indicate there is no more insight gained from CC when compared to LOC.

The relationship between CC and LOC is near linear regardless of language type for the three languages in this study. This result implies that the characteristic of complexity and test case size measured by CC and LOC is the same in a procedural language (C), an objected-oriented language (Java) and a hybrid language (C++). It also implies that if CC indeed measures some aspects of complexity, then developers tend to add these aspects to a program at an incredibly steady rate (at least in practice).

Modules where LOC does not predict CC are outliers and should be targeted for closer scrutiny. These models on average accounted for 90% of CC's variance. This means that any source-file/program which does not fit this model is in a statistical sense an outlier. If the outlier status of these modules to the model is equally (or even partially) indicative of a similar status for *true* complexity then these linear models themselves can be used as a form of complexity metric or at least as a monitor for possible complexity issues. Modules where LOC does not predict CC (or vice-versa) may indicate an overly-complex module with a high density of decision points or

an overly-simple module that may need to be refactored. We plan to pursue this line of inquiry in future work.

7. Conclusions

We carried out a large empirical study of the relationship between LOC and CC for a sample population that crossed languages, methodologies, and programming paradigms. We found that due mostly to issues regarding population variance, that the linearity of the relationship between these two measurements has been severely underestimated. Using modern statistical tools we develop linear models that can account for the majority of CC by LOC alone. We conclude that CC has no explanatory power of its own and that LOC and CC measure the same property. We also conclude that if CC does have any validity as a measure of either complexity or test space size, then we must conclude these factors grow linearly with size regardless of software language, paradigm, or methodology. The stability of the linear relationships we found suggests future work in examining their worth as metrics in their own right.

REFERENCES

[1] R. D. Banker, M. D. Srikant, C. F. Kemerer, and D. Zweig, "Software complexity and maintenance cost," Communications of the ACM, Vol. 36, No. 11, pp. 81–94, 1993.

[2] G. K. Gill and C. F. Kemerer, "Cyclomatic complexity density and software maintenance productivity," IEEE Transactions on Software Engineering, Vol. 17, No.12, pp. 1284–1288, 1991. (REF 15)

[3] F. G. Wilkie and B. Hylands, "Measuring complexity in C++ application software," Software: Practice and Experience, Vol. 28, No. 5, pp. 513–546, 1998. (REF 17)

[4] B. Curtis, S. B. Sheppard and P. Milliman, "Third time charm: Stronger prediction of programmer performance by software complexity metrics," Proceedings of the 4th International Conference on Software Engineering, pp. 356–360, 1979.

[5] J. C. Munson and T. M. Khoshgoftaar. "The detection of fault-prone programs," IEEE Transactions on Software Engineering, Vol. 18, No. 5, pp. 423–433, 1992.

[6] V. R. Basili and B. T. Perricone, "Software errors and complexity: An empirical investigation," Communications of the ACM, Vol. 27, No. 1, pp. 42–52. 1983. (REF 4)

[7] J. C. Munsona and T. M. Khoshgoftaar, "The dimensionality of program complexity," Proceedings of the 11th International Conference on Software Engineering, pp. 245–253, 1989.

[8] T. J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, Vol. 2, No. 4, pp. 308–320, 1976.

[9] M. Shepperd, "A critique of cyclomatic complexity as a software metric," Software Engineering Journal, Vol. 3, No. 2, pp. 30–36, 1988.

[10] A. R. Feuer and E. B. Fowlkes, "Some results from an empirical study of computer software," Proceedings of the 4th International Conference on Software Engineering, pp. 351–355, 1979. (REF 6)

[11] R. Subramanyam and M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects," IEEE Transactions on Software Engineering, Vol. 29, No. 4, pp. 297–310. 2003.

[12] W. Li, "An empirical study of software reuse in reconstructive maintenance," Software Maintenance: Research and Practice, Vol. 9, pp. 69–83, 1997.

[13] S. J. Sayyad and T. J. Menzies, The PROMISE Repository of Software Engineering Databases, School of Information Technology and Engineering, University of Ottawa, Canada, http://promise.site.uottawa.ca/SERepository.

[14] SourceForge, http://www.sourceforge.net/.

[15] Alexa, Top 500, 2007, http://www.alexa.com/.

[16] Hewlett Packard: HP-sponsored projects hosted on SourceForge, 2007, http://hp.sourceforge.net/.

[17] IBM, http://sourceforge.net/powerbar/websphere/.

[18] Subversion, http://subversion.tigris.org/.

[19] RSM, http://www.msquaredtechnologies.com/.

[20] CCCC, C and C++ Code Counter, 2007, http://sourceforge.net/projects/cccc/.

[21] T. S. Breusch and A. R. Pagan, "A simple test for heteroscedasticity and random coefficient variation," Econometrica, Vol. 47, No. 5, pp. 1287–1294, 1979.

[22] A. F. Siegel, "Robust regression using repeated medians," Biometrika, Vol. 69, No. 1, pp. 242–244, 1982.

[23] H. Zuse, "A framework of software measurement," Walter de Gruyter, New York, 1998.

[24] J. Bowen, "Are current approaches sufficient for measuring software quality?" Proceedings of Software Quality Assurance Workshop, pp. 148–155, 1978.

[25] M. R. Woodward, M. A. Hennell and D. A. Hedley, "A measure of control flow complexity in program text," IEEE Transactions on Software Engineering, Vol. 5, No. 1, pp. 45–50, 1979.

[26] M. Paige, "A metric for software test planning," Proceedings of COMPSAC 80 Conference, Buffalo, NY, pp. 499–504, Oct. 1980.

[27] T. Menzies, J. Greenwald and A. Frank. "Data mining static code attributes to learn defect predictors," IEEE Transactions on Software Engineering, Vol. 33, No. 1, 2–13, 2007.

J. Software Engineering & Applications, **2009, 2: 144-149**
doi:10.4236/jsea.2009.23021 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Scientific Research

Applying Heuristic Search for Distributed Software Performance Enhancement

Omid BUSHEHRIAN

Computer and IT Department, Shiraz University of Technology, Shiraz, Iran.
Email: bushehrian@sutech.ac.ir

Received April 2nd, 2009; revised April 28th, 2009; accepted May 4th, 2009.

ABSTRACT

Software reverse engineering and reengineering techniques are most often applied to reconstruct the software architecture with respect to quality constraints, or non-functional requirements such as maintainability or reusability. In this paper, the performance improvement of distributed software is modeled as a search problem that is solved by heuristic search algorithms such as genetic search methods. To achieve this, firstly, all aspects of the distributed execution of a software is specified by an analytical performance evaluation function that not only evaluates the current deployment of the software from the performance perspective but also can be applied to propose the near-optimal object deployment for that software. This analytical function is applied as the Heuristic search objective function. In this paper a novel statement reordering method is also presented which is used to generate the search objective function such that the best solution in the search space can be found.

Keywords*: *Performance Engineering, Heuristic Search Methods, Software Reverse Engineering

1. Introduction

Automatic software reverse engineering and reengineering techniques are most often applied to reconstruct the software architecture with respect to quality constraints, or non-functional requirements such as maintainability or reusability [1–5]. However, there has been no effort to assess the architecture of existing distributed software from the performance viewpoint. All the architectural level performance engineering techniques are focused on the performance assessment at the early stages of the software development life cycle. However, the implemented software still may not meet its performance provisions and needs to be modified to improve the performance.

In this paper, a novel software reengineering approach is presented that proposes some alterations to the deployment of the distributed software to improve its overall performance. The performance improvement is achieved by providing the chance for concurrent execution among method calls including the remote calls or local ones. The concurrency among the method calls is obtained when some of the blocking invocations are transformed into non-blocking or asynchronous form. However, this transformation entails execution overheads that should be considered very carefully. Therefore, the question is how to automatically find a set of invocations to be transformed to non-blocking such that the highest amount of concurrency in the execution of distributed software over a cluster is obtained.

To address this problem, in this paper the program source code is analyzed to extract a performance evaluation function considering the characteristics of the current deployment of the distributed software. These characteristics include the number of available workstations, the number of processors of each workstation and the deployment constraints. According to these constraints, some of the objects must reside on specific workstations as they need to access hardware or software resources (such as Database, printer, file,…) on that workstation.

The current researches in the Software Performance Engineering (S.P.E) are dedicated to estimate the performance of software in the early stage of development process due to needs for *QOS*. In order to achieve this goal, several works have been done to transform the software architectural models to the analyzable formal models. Some examples are deriving Queuing Network models from UML diagrams [6] or translating some of the UML diagrams to the Perti Nets [7,8]. In [9] constructing and analyzing two kinds of performance models are proposed: *software execution model* and *system execution model*. The former represents the software execution behavior and is modeled by *execution graphs* and the latter is based on the queuing network models, which

represent the computer system platform, including hardware and software components. The software and system execution models are applied to assess the performance of the intended software architecture.

There are also some related researches in the area of performance optimization of existing distributed programs. In a mixed dynamic and programmer-driven approach to optimize the performance of large-scale object-oriented distributed systems [10], an object-partitioning method is dynamically invoked at runtime to collocate objects that communicate often. Here, the partitioning criterion is to gather objects that often communicate in a same partition. In a distribution strategy for program objects [11], an object graph construction algorithm and a partitioning policy for the program objects based on object types is presented. The distribution strategy is to place the most communicating objects in the same machine to minimize the network traffic. However, when partitioning a program, in addition to minimizing the communication cost, the amount of concurrency in the execution of the distributed partitions has to be maximized. To achieve this, in this paper, a new performance-driven partitioning criterion is proposed.

The main contribution of this paper is to extend the Software Performance Engineering techniques to the reengineering area in order to optimize the performance of existing distributed software. To achieve this, a new parametric performance evaluation function to assess the performance of the current object deployment of the software over the computational nodes is presented. This function not only evaluates the current software deployment but also proposes the best object deployment for the software. This function is automatically constructed while traversing the program call flow graph and considers both blocking and non-blocking types for each invocation.

The remaining parts of this paper are organized as follows: in section 2 the main steps of the proposed method are described. Section 3 presents an optimization technique called *statement reordering* which is applied to improve the amount of concurrency in the distributed program code. In section 4 the implementation of the proposed method is described. Finally the conclusions and future works are presented in section 5.

2. Software Performance Modeling

To improve the performance of distributed software, some of the blocking invocations among objects should be transformed to non-blocking ones. The non-blocking invocations are implemented either by means of remote asynchronous calls (supported in some middlewares such as JavaSymphoney [12]) or Java Threads. However, an invocation is converted to non-blocking only when this conversion results in concurrent execution between the caller and callee considering the resultant overheads. Each non-blocking invocation incurs communication and initiation overheads. The former is the amount of required time for sending the invocation parameters and receiving the return values between caller and calee. The latter is the amount of time required for starting the non-blocking invocation (such as asynchronous RMI or Java Threads).

The object deployment for a given program is defined as pair *(r,d)*. *r* denotes the set of all invocations in the program along with each invocation status (blocking or non blocking) an *d* denotes the deployment of caller and called objects, for each invocation I_i in *r*, over the available computational nodes. The performance evaluation function $\Theta(r,d)$ estimates the amount of execution time for object deployment (r,d). The optimal object deployment (r_o,d_o) is the one for which the amount of Θ is minimum. To find the optimal object deployment all possible object deployment for the software should be evaluated using function Θ. However, this is a NP-Complete problem and should be solved using heuristic search algorithms such as Genetic algorithms. The deployment constraints must be considered during the search for the optimal object distribution. According to these constraints, some of the objects must reside on specific workstations as they need to access hardware or software resources (such as Database, printer, file,…) on that workstation. We have used a Constrained Genetic Clustering algorithm to find the optimal object deployment. In this algorithm function Θ is used as the search objective function.

The main steps can be summarized as follows:

- Extracting *Call Graph* from the program source code.
- Unfolding the all graph to obtain the *Call Tree.*
- Extracting function Θ by analyzing the program source code.
- Search for the optimal object deployment (r_o,d_o). This is achieved by using a genetic clustering algorithm that uses Θ as the search objective function.

We have omitted the cycles in the extracted call graph to obtain the program call tree. This is necessary as in the call graph each node represents a class with multiple incoming invocations. However at runtime each invocation may be performed using a separate instance of a class. Each object deployment (r,d) corresponds to a *labeled partitioning* of the program call tree. Each label of a node in the call tree specifies the workstation on which the object represented by that node resides. The status of invocations is determined by the partitioning. The invocations among partitions are assumed non-blocking while the invocations inside a partition are blocking. Therefore each labeled partitioning of the call tree specifies an object deployment (r,d) of the software uniquely and vice versa. To search for the optimal object deployment (r_o,d_o) all possible labeled partitioning of the call tree are evalu-

ated by the constrained genetic clustering algorithm.

The performance evaluation function Θ is built automatically while traversing a program call flow graph. A call flow graph represents the flow of method calls among program classes. Θ is built considering the estimated execution times of all instructions within the program code including the invocations. Since, the type of invocations affects the overall execution time of the program and is not determined until the program is partitioned; Θ is built as a general function including all the possible invocation types for each method call. There are four types of invocations: local blocking, remote blocking (implemented using RMI), local non-blocking (implemented using Java Thread) and remote non-blocking (implemented using asynchronous RMI). For each type of invocation an overhead is defined as shown in Table 1.

As described in the previous section, Θ maps each object distribution (*r,d*) to a value representing the estimated execution time of the distributed software with object distribution (*r,d*). Object distribution (*r,d*) is defined using a set of functions presented in Table 2 .

For instance, consider a method invocation I_1 that performs another invocation I_2 during its execution. The estimated execution time of I_1, denoted by T_{I1}, when I_2 type is (1) local blocking, (2) remote blocking,(3) local non-blocking and (4) remote non-blocking , is shown by relations (1) to (4) below respectively.

Table1. Different overhead types

α	RMI initiation overhead
β	Thread creation overhead
γ	Asynch RMI initiation overhead

Table 2. The maps that specify a *labeled portioning* and its equivalent object deployment (*r,d*)

Φ	Maps invocation I_i to a value 0 or 1, if I_i is inside a partition it is 1 otherwise it is 0.
μ	Maps invocation I_i to a value 0 or 1, if the caller and callee objects of I_i reside on the same workstation it is 0, otherwise it is 1.
ω	Maps invocation I_i to a workstation name on which Ii is executed
Δ	Maps invocation I_i to a value indicating the communication cost of the network link over which I_i is sending parameters or receiving return values as a RMI or asynchronous RMI
Γ	Maps each workstation w to the maximum number of threads created on it
Π	Maps each workstation w to the number of processing units installed on it

$$T_{I1} = T_0 + T_{I2}, \quad T_{I2}=T_1 \tag{1}$$

$$T_{I1} = T_0 + \alpha + T_{I2} + \delta(I_2), \quad T_{I2}=T_1 \tag{2}$$

$$T_{I1} = T_0 + \beta + S_2, \quad S_2=\max(T_1-d_2,0) \tag{3}$$

$$T_{I1} = T_0 + \gamma + S_2, \quad S_2=\max(T_1+\delta(I_2)-d_2,0) \tag{4}$$

Assuming that the target of the invocation I_1 is method *m*, T_0 is the total execution time of all the instructions excluding I_2 within *m*. When an invocation such as I_2 is non-blocking, the caller should wait for the results of I_2 at some synchronization point during its execution. In relation (3) and (4), S_2 denotes the amount of required time at the synchronization point of I_2, to receive the results of I_2. The above relations can be combined as a single relation as follows:

$$\begin{aligned} T_{I1} &= T_0+\Phi(I_2)*(T_{I2}+\mu(I_2)*(\alpha+\delta(I_2))) \\ &\quad + (1-\Phi(I_2))*(S_2+\mu(I_2)*\gamma+(1-\mu(I_2))*\beta) \\ S_2 &= \max(T_1+\mu(I_2)*\delta(I_2)-d_2,0) \end{aligned} \tag{5}$$

There may be several invocations, I_i, within method *m* and each invocation itself may include other invocations. Therefore, relation (5) for estimating the execution time of I_1 can be generalized as follows:

$$\begin{aligned} T_{I1} &= T_0 + \sum \Phi(I_i)*(T_{Ii} + \mu(I_i)*(\alpha+\delta(I_i))) \\ &\quad +\sum (1-\Phi(I_i))*(S_i + \mu(I_i)*\gamma+(1-\mu(I_i))*\beta) \\ S_i &= \max(T_{Ii}+\mu(I_i)*\delta(I_i) - d_i,0) \end{aligned} \tag{6}$$

In the above relation, S_i denotes the time elapsed to wait for the results of the invocation I_i, d_i denotes the estimated execution time of the program statements located between each call statement, I_i, and the first locations where the results of the call are required (the synchronization point of I_i).

We can generalize relation (6) to obtain function Θ. this function that returns the estimated execution time for object deployment (r,d) is built by applying relations (6) recursively starting from the *main*() method of the program. Assuming that the program call flow graph is cycle-free, Θ (*r,d*) can always be computed recursively. However, there may be cycles in the call flow graph, resulting from direct or indirect recursive calls. Assuming that I_i is an invocation to a method in the cycle (and itself is not in the cycle) and the estimated number of recursions is n_i then the estimated execution time of invocation I_i is multiplied by n_i. An invocation I_i or a synchronization point S_i may be located within a loop statement. Therefore to consider the impact of loop iterations on the time estimation, coefficients m_i and k_i have been added to relation (7):

$$\begin{aligned} \Theta_{main}(r,d) &= T_0 +\sum\Phi(I_i)*n_i*m_i*(\Theta_{Ii}(r,d) \\ &\quad +\mu(I_i)*(\alpha+\delta(I_i)))+\sum (1-\Phi(I_i))* \\ &\quad (S_i + \mu(I_i)*\gamma+(1-\mu(I_i))*\beta) \\ S_i &= k_i*\max(H(I_i)*\Theta_{Ii}(r,d)+\mu(I_i)*\delta(I_i) - d_i,0) \end{aligned} \tag{7}$$

In the above relation, $\Theta_{Ii}(r,d)$ denotes the estimated execution time of the program starting from invocation I_i. $H(I_i)$ is the adjustment factor that adjusts the execution time of invocation I_i according to the hardware capacity of the workstation on which I_i will be executed. $H(I_i)$ is defined as follows:

$$H(I_i)= \Gamma\ (\omega(I_i))\ /\ \Pi(\omega(I_i)) \quad (8)$$

As described in Table 2, Γ maps each workstation w to the number of possible threads created on it due to remote or local non-blocking invocations executed on w considering an object distribution (r,d). $\Gamma(w)$ for each workstation *w* in object distribution (r,d) is calculated as follows:

$$\Gamma(w)=\sum\ \Phi(I_i)*\mu(I_i)+(1-\ \Phi(I_i))\ , \quad (9)$$

for all I_i such that: $\omega(I_i)=w$

3. Statement Reordering

In the preceding sections, the idea of partitioning the program call tree directed by Θ function was described. The main idea was to search for a partitioning of the program classes which results in the highest amount of concurrency among invocations. Obviously the concurrency is achieved by performing some of the method invocations among actors asynchronously. Generally, converting an ordinary method call to an asynchronous one, poses two kinds of overheads on the execution: initiation and communication (described earlier). The former is denoted by *s* and the latter is denoted by *O*. Therefore, from the performance perspective, converting an ordinary call I_k to an asynchronous call is only beneficial when the sum of execution times of program instructions located between I_i and its synchronization point S_i , denoted by d_i, is greater than $s+O$. Obviously, the larger the amount of d_i, the more concurrency between the caller and the callee is obtained.

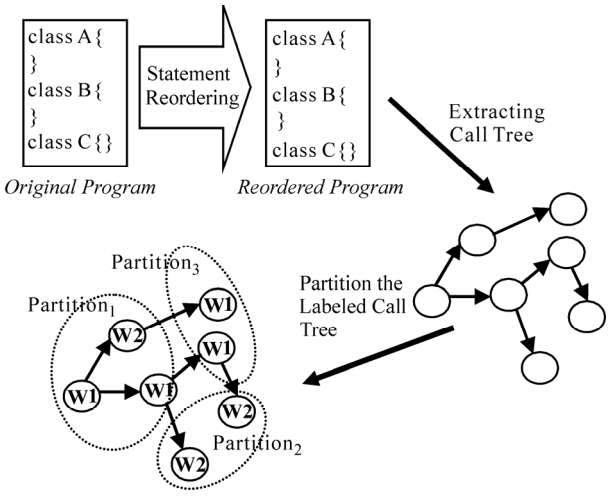


Figure 1. The statement reordering

However, a major difficulty is that programmers usually use the results of any method call I_i immediately after (d_i=0). Therefore there will be no chance for concurrency when transforming method calls to asynchronous calls. To resolve the difficulty, we have applied the ideas of *statement reordering* to enhance the potential parallelism degree of a program by increasing the amount of d_i for each invocation. The algorithm attempts to insert as many statements as possible between an invocation and its first data-dependent statement, considering the data dependencies between the statements. Obviously the reordering should be performed such that the original semantic of the program is preserved. To do this during the statement reordering the data and control dependency among statements must not be violated. Data and control dependency among program statements are represented by a acyclic graph called *Task Graph* [13]. The statement reordering is performed such that the dependencies represented by the program task graph are not violated. The overall steps including statement reordering is illustrated in Figure 1.

In the statement reordering algorithm, the program statements are classified as follows:

- *Call*: a method invocation
- *Use*, statement which is data dependent on a *Call*
- *Common*, an ordinary statement which is neither a *Call* nor a *Use*

The algorithm comprises two main steps. In the first step the program statements are moved from the *Original Code* to the *Reordered Code* gradually. In this step, presented in Figure 2, *Call* statements are moved to the reordered cod first and *Use* statements are moved as late as possible. In the second step, the reordered code resulted by applying the first step, is further optimized by reducing the time elapsed to wait for the results of *Call* statements. This is achieved by inserting as many statements as possible between each *Call* and its corresponding *Use*.

In the first step, *Call* statements of longer execution time are moved to the reordered code first because the longer the execution time of a *Call*, the more statements should be inserted between that *Call* and the corresponding *Use*. Obviously, before a statement is moved into the reordered code, all of its parent statements in the program task graph should be moved into the reordered code.

In the second step, the reordered code resulted in the first step is further optimized. To achieve this, the time required to wait for the results of *Call* statements is minimized. This is achieved by pushing down *Use* statements with positive wait time as far as their wait time reaches zero. Here Use statements with longer wait time are selected and pushed down first.

4. Implementation

We have developed a tool support that implements the

steps described in section 2 to find the optimal object distribution for distributed software. Within this environment, a Java source code analyzer implemented using COMPOST [14] library, analyzes the program source to extract the call graph, call flow graph, call tree and build the Θ function. To build the Θ function for a program, first the number of clock cycles for each statement in the program is determined, according to the JOP microinstruction definition [15], and saved in an XML document. JOP is an implementation of Java Virtual Machine in hardware. Our tool also inputs the loop bounds in the program as they are needed during the Θ generation. The generated XML document is applied by a statement reordering [13] engine to maximize the distances between each call statement and its very first data-dependent statement. The reordered program and the XML document are input to a separate module to produce the Θ function. The resultant Θ function is applied as an objective function of a constrained genetic clustering algorithm [16] to find a near-optimal object deployment for the distributed software.

Algorithm Reorder (Task-Graph: DAG) OUT: Reordered-Code : List
Begin
1. If there is no *Call* in Task-Graph which is not *moved*
2. While there is a *Common-statement* in Task-Graph which is *not moved*
3. Select a *Common-statement* whose predecessors in Task-Graph are already *moved*
4. Append this statement to Reordered-Code
5. Label this instruction as *moved*
6. End While
7. While there is a *Use* in Task-Graph which is not *moved*
8. Select a *Use* whose predecessors in Task-Graph are already *moved*
9. Append this instruction to Reordered-Code
10. Label this instruction as *moved*
11. End While
12. Else
13. Find a *Call* C with the longest execution time in Task-Graph which is *not moved*
14. Let New-Task-Graph be a Sub-graph of Task-Graph, including Predecessors of C
15. Reorder(New-Task-Graph)
16. Append C to Reordered-Code
17. Label C as *moved*
18. Reorder(Task-Graph)
19. End If
End

Algorithm Optimize (Task-Graph: DAG, Reordered-Code : List) OUT: New-Reordered-Code: List
Begin
1. While there is a *Use* in Task-Graph which is not selected
2. Select a *Use* U with the longest wait time W in Task-Graph
3. Find all unselected nodes in Task-Graph which are not connected to U through a path in the task graph and
4. Add them to *Moving-List*
5. Remove all those nodes from *Moving-List* which do not share an immediate control dependent parent with U
6. While W>0
7. Select an instruction I, from *Moving-List*, whose predecessors in Task-Graph are not in *Moving-List*
8. Let P be the set of immediate predecessors of I
9. Let C be a *Call* whose results are used by U
10. Let P=P ∪ {C}
11. Move I to a position after instructions in P and before U in Reordered-Code
12. Remove I from *Moving-List*
13. Subtract the estimated time of I form W
End While
End While
End

Figure 2. The statement reordering algorithms

A practical evaluation of the proposed method to optimize the performance of distributed object-oriented programs is presented in this section. We have used two Java case-studies: *TSP* and *Consolidated Clustering (CC)*. The first case study evaluates the impact of applying the proposed approach on a TSP program containing 18 classes and 129 method calls. This program finds near optimal Hamiltonian Circuit in a graph, using minimum spanning trees. The second case study measures the amount of speedup achieved by optimizing the performance of a program called Consolidated Clustering [17]. Consolidated Clustering is a graph clustering application written in Java. This program comprises 16 classes and 23 method calls. In this program, a graph is clustered several times using heuristic clustering algorithms. The results of each clustering are stored in a database for further uses. This program consolidates the clustering results to obtain a clustering with a specific confidence threshold. The program is relatively slow, because it applies the heuristic algorithms for clustering.

The case studies were analyzed first to extract function Θ for them. Then a genetic clustering algorithm was applied to partition the call tree of each program to find the optimal object deployment using Θ as the objective function. The chosen test bed was a cluster with 3 single processor Pentium computers running *JavaSymphoney* [12] as the cluster middleware. The parameter passing mechanism in remote non-blocking invocations in this test bed is implemented using copy-restore technique.

Before applying the genetic clustering algorithm on the case-studies, the values for parameters α, β, γ and $\delta(I_i)$ were measured in the test bed. The amount of communication cost over the Pentium cluster, regarding our underlying communication middleware, *JavaSymphony*, was measured less than 100 ms. To estimate the communication cost in terms of JOP clock cycles, the number of clock cycles of a sample program was divided by the measured execution time of that program. According to this estimation, the communication cost $\delta(I_i)$ was nearly 10^7 clock cycles for all I_i. The measured amount of parameters α, β and γ in the test bed were almost the same and was estimated 10^4 clock cycles. We applied a constraint for the CC program as 3 of its classes in the call tree should necessarily reside on the workstation on

Table 3. Measured speedups

	20	40	100	130	400	500
TSP:Θ	0.8	0.9	1	1.2	1.35	1.4
CC: Θ	0.6	0.7	0.8	1	1.3	1.4
TSP:MCC	0.3	0.5	0.6	0.7	0.78	0.8
TSP:TM	0.6	0.8	0.2	0.8	1	0.8

which the Data Base server was running.

The measured speedups resulted from executing TSP and CC on the test bed are presented in Table 3. The results are compared to other methods: (1) applying *MCC* function and (2) and applying trivial method. MCC denotes *Minimum Communication Cost* described in section 1. In the trivial method, denoted by TM, the invocations are assigned to a workstation with the lowest load at runtime for execution.

5. Conclusions

In this paper the software reengineering research area has been extended to include performance improvement of existing distributed software. To achieve this first a performance assessment function is extracted from the program source code. Then this function is applied to find optimal object deployment of the software using a constrained genetic clustering algorithm. The result is a labeled partitioning of the program call tree. The invocations inside a partition are assumed blocking while the invocations among partitions are non-blocking. The labels in the labeled partitioning graph indicate the workstations on which objects reside. We have implemented this approach and applied that on two case studies. The result of our measurements shows this approach can be applied to improve the performance of legacy software.

This is an ongoing research in the field of Software Performance Engineering. As the future work we intend to extend the idea of architectural level performance assessment in forward engineering to validate the software models in the sense that whether they satisfy the performance provisions or not.

REFERENCES

[1] B. Bellay and Gallh, "Reverse engineering to recover and describe a systems architecture," Development and Evolution of Software Architectures for Product Families Lecture Notes in Computer Science, Vol. 1429, 1998.

[2] D. R. Harris, H. B. Reubenstein and A.S.Yeh, "Reverse engineering to the architectural level," Proc. 17th Int. Conf. Software Engineering, Seattle, Washington, US, 1995.

[3] S. Parsa and O. Bushehrian, "The design and implementation of a tool for automatic software modularization," J. Supercomput., Vol. 32, No. 1, pp. 71–94, 2005.

[4] B. S. Mitchell and M. Spiros, "Bunch: A clustering tool for the recovery and maintenance of software system structure," Proc. Int. Conf. Software Maintenance, 1999. (IEEE)

[5] L. Tahvildari, K. Kontoglannis and J. Mylopoulos, "Qualitydriven software re-engineering," J. Syst. Softw., Vol. 66, pp. 225–239, 2003.

[6] Hyunsang Youn, Suhyeon Jang and Eunseok Lee, "Deriving queuing network model for UML for software performance prediction," Fifth International Conference on Software Engineering Research, Management and Application, pp. 125–131, 2007. (IEEE)

[7] J. M. Fernandes, S. Tjell, J. B. Jorgensen and O. R. Ribeiro, "Designing tool support for translating use cases and UML 2.0 sequence diagrams into a colored Petri Net," Proc. 16th international Workshop on Scenarios and State Machines, 2007. (IEEE)

[8] R. G. Pettit and H. Gomma, "Analyzing behavior of concurrent software designs for embedded systems," Proc. 10th International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2007. (IEEE)

[9] Andolfif., F. Aquilani, S. Balsamo, and P. Inverardi, "Deriving performance models of software architectures from message sequence charts," Proc. 2nd Int. Workshop on Software and Performance (WOSP2000), Canada, 2000.

[10] Y. Gourhant, S. Louboutin, V. Cahill, A. Condon, G. Starovic, and B. Tangney, "Dynamic clustering in an object-oriented distributed system," Proc. OLDA-II (Objects in Large Distributed Applications), Ottawa, Canada, October 1992.

[11] D. Deb, M. Fuad, and M. J. Oudshoom, "Towards autonomic distribution of existing object oriented programs," Int. Conf. Autonomic and Autonomous Systems (ICAS' 06), 2006. (IEEE)

[12] T. Fahringer and A. Jugravu, "JavaSymphony: New directives to control and synchronize locality, parallelism, and load balancing for cluster and GRID-computing," Proc. Joint ACM Java Grande–ISCOPE 2002 Conf., Seattle, Washington, November 2002.

[13] S. Parsa and O. Bushehrian, "Genetic clustering with constraints," Journal of research and practice in information technology, Vol. 39, No. 1, pp. 47–60, 2007.

[14] http://www.info.uni-karlsruhe.de/~compost, last visit: 12th September 2009.

[15] M. Schoeberl, "A time predictable Java processor," Proc. Conf. Design, Automation and Test in Europe, Germany, pp. 800–805, 2006.

[16] S. Parsa and O. Bushehrian, "Performance-driven object oriented program re-modularization," Journal of IET Software, Vol. 2, No. 4, pp. 362–378, 2008.

[17] B. S. Mitchell, "A heuristic search approach to solving the software clustering problem," Ph.D Thesis, Drexel University, March 2002.

J. Software Engineering & Applications, 2009, 2: 150-159
doi:10.4236/jsea.2009.23022 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Data Mining in Biomedicine: Current Applications and Further Directions for Research

S. L. TING[1], C. C. SHUM[2], S. K. KWOK[1], A. H. C. TSANG[1], W. B. LEE[1]

[1]Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China; [2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China.
Email: jacky.ting@polyu.edu.hk

Received January 16th, 2009; revised June 18th, 2009; accepted June 24th, 2009.

ABSTRACT

Data mining is the process of finding the patterns, associations or relationships among data using different analytical techniques involving the creation of a model and the concluded result will become useful information or knowledge. The advancement of the new medical deceives and the database management systems create a huge number of databases in the biomedicine world. Establishing a methodology for knowledge discovery and management of the large amounts of heterogeneous data has become a major priority of research. This paper introduces some basic data mining techniques, unsupervised learning and supervising learning, and reviews the application of data mining in biomedicine. Applications of the multimedia mining, including text, image, video and web mining are discussed. The key issues faced by the computing professional, medical doctors and clinicians are highlighted. We also state some foreseeable future developments in the field. Although extracting useful information from raw biomedical data is a challenging task, data mining is still a good area of scientific study and remains a promising and rich field for research.

Keywords*: *Data Mining, Biomedicine

1. Introduction

With the tremendous improvement in the speed of computer and the decreasing cost of data storage, huge volumes of data are created. However, data itself has no value. Only if data can be changed to information, it becomes useful. In order to generate meaningful information, or knowledge from database, the field of data mining was born. The data mining field is about two decade old. Early pioneers such as U. Fayyad, H. Mannila, G. Piatetsky-Shapiro, G. Djorgovski, W. Frawley, P. Smith, and others found that the traditional statistical techniques were not adequate to handle the mass amount of data. They recognized the need of better, faster and cheaper ways to deal with the dramatic increase in the amount of data.

Nowadays, besides the numerous number of databases created and accumulated in a dramatic speed, data is no longer restricted to numeric or character only especially in the biomedicine aspect. The advanced medical deceives and database management systems enable the integration of the different types of high dimensional multimedia data (e.g. text, image, audio, and video) under the same umbrella. Establishing a methodology for knowledge discovery and management of large amounts of heterogeneous data has therefore become a main priority. Various techniques are used in different areas of biomedicine, including genomics, proteomics, medical diagnosis, effective drug design and pharmaceutical industry.

In this paper, we would first give a brief outline on what is data mining, its position or role in the knowledge discovery process and the basic principles of some commonly used data mining techniques. Next, we present our investigation results of the applications of the data mining in the biomedicine aspect, which includes the area of biology, medicine, pharmacy and health care. Lastly, we discuss some difficulties of data mining in biomedicine and the possible direction for the future development.

2. What is Data Mining?

Data mining (DM) is the process of finding the patterns, associations or relationships among data using different analytical techniques involving the creation of a model and the concluded result will become useful information or knowledge. DM can also be expressed as

- Nontrivial extraction of implicit, previously unknown, and potentially useful information from data [1]; and
- Making sense of large amounts of mostly unsupervised data in some domain [2]

It is an interdisciplinary subject that lies at the inter face of pattern recognition and database systems and emerges the techniques from the mathematics and statistical disciplines as well as from the artificial intelligence and machine leaning communities. It has a great deal in common with statistics but on the other hand, there are differences. Unlike statistics, data mining can be due with heterogeneous data fields.

Very often, the term knowledge discovery is used together with Data Mining. Knowledge discovery, also known as knowledge discovery in database (KDD), is the process that seeks new knowledge in some application domain. DM is one of the steps in the knowledge discovery process. Figure 1 is an outline of the six step hybrid KDD model developed by [2].

The initial step of understanding the problem domain involves working closely with domain experts to define the problem and determine the project goals, and learning about current solutions to the problem. A description of the problem, including its restrictions, is prepared. The DM tool to be used in the later stage is selected. Next, we need to understand the data which includes collecting sample data and deciding which data, including format and size, will be needed. Data are checked for completeness, redundancy, missing values, plausibility of attribute values, etc. Preparation of data decides which data will be used as input for DM methods in the subsequent step. It involves sampling, running correlation and significance tests, and data cleaning. Data miner then uses various DM methods to derive knowledge from preprocessed data. Evaluation includes understanding and checking if the result is novel. Finally, we will decide how to use and deploy the discovered knowledge.

3. Data Mining Techniques

Data mining techniques fall into two broad categories: unsupervised and supervised. Unsupervised learning refers to the technique that is not guided by any particular variable or class label. In the unsupervised learning, we do not create a model or hypothesis prior to the analysis. We apply the algorithm directly to the data and observe the results. A model will then be built according to the results. Thus, unsupervised leaning is used to define class for data without class assignments. Clustering is one of the common unsupervised techniques.

In contrast, for supervised learning, a model is built prior to the analysis. We then apply the algorithm to the data in order to estimate the parameters of the model. The objective of building models using supervised learning is to predict an outcome or category of interest. The biomedical literature on applications of supervised learning techniques is vast. Classification, statistical regression and association rules building are very common supervised learning techniques used in medical and clinical research. Table 1 is the summary comparing the characteristics and the techniques used for the two different learning methods. Followed is a brief explanation

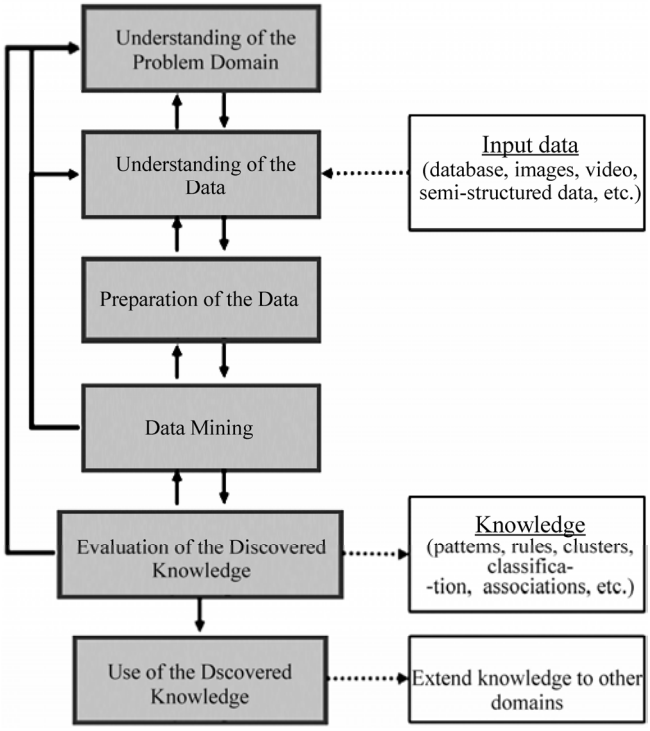


Figure 1. Six-step hybrid KDD model [2]

Table 1. Comparing the characteristics and the techniques of the unsupervised and supervised learning

	Characteristics	Techniques
Unsupervised Learning	• No guidance • Use to Define the class • Seldom utilized (until recently)	• Clustering • Association Rule
Supervised Learning	• With guidelines • Class defined • Common with vast literature and application	• Classification • Statistical Regression • Artificial neural networks

of the four learning techniques.

3.1 Clustering

Clustering is an unsupervised learning technique revealing natural groupings in the data. Cluster analysis refers to the grouping of a set of data objects into clusters. A cluster is a collection of data objects which are similar to one another within the same cluster but not similar to the objects in another cluster. Clustering is also called unsupervised classification where no predefined classes are assigned.

3.2 Association Rule

Association rule discovery is to find the relationships between the different items in a data base. It is normally express in the form X => Y, where X and Y are sets of attributes of the dataset which implies that transactions that contain X also contain Y.

3.3 Classification

Classification is a supervised learning method. It is a method of categorizing or assigning class labels to a pattern set under the supervision. The object of classification is to develop a model for each class. Classification methods can usually be categorized as follows:

a) Decision tree

Decision tree classifiers divide a decision space into piecewise constant regions. It splits a dataset on the basis of discrete decisions, using certain thresholds on the attribute values. It is one of the most widely used classification method as it is easy to interpret and can be represented under the If-then-else rule condition.

b) Nearest-neighbor

Nearest-neighbor classifiers [3] typically define the proximity between instances, find the neighbors if a new instance, and then assign to it the label for the majority class of its neighbors.

c) Probabilistic models

Probabilistic models are models which calculate probabilities for hypotheses base on Bayes' theorem [3].

3.4 Statistical Regression

Regression models are very popular in the biomedical literature and have been applied in virtually every subspecialty of medical research. Before computers were widely used, linear regression was the most popular model to find solutions of the problem of estimating the intercept and coefficients of the regression question. It has solid foundation from the statistical theory. Linear regression is similar to the task of finding the line that minimizes the total distance to a set of data. That is find the equation for line $Y = a + bX$. With the help of computers and software package, we can calculate the high complex models.

3.5 Artificial Neural Networks

Artificial neural networks [4] are signal processing systems that try to emulate the behavior of human brain by providing a mathematical model of combination of numerous neurons connected in a network. It learns through examples and discriminate the characteristics among various pattern classes by reducing the error and automatically discovering inherent relationships in a data-rich environment. No rules or programmed information is need beforehand. It composes of many elements, called nodes which are connected in between. The connection between two nodes is weighted and by the adjustment of these weights, the training of the network is performed. The weights are network parameters and their values are obtained after the training procedure. There are usually several layers of nodes. During the training procedure, the inputs are directed in the input layer with the desirable output values as targets. A comparison mechanism will operates between the out and the target value and the weights are adjusted in order to reduce error. The procedure is repeated until the network output matches the targets. There are many advantages of neural networks like adaptive learning ability, self-organization, real-time operation and insensitivity to noise. However, it also has a huge disadvantage that it is highly dependence on the training data and it does not provide an explanation for the decisions they make, just like working in the 'black box'.

3.6 Advanced Data Mining Techniques

During the past few years, researchers have tried to combine both unsupervised and supervised methods for the

analysis [5]. Some examples of advanced unsupervised learning models are hierarchical clustering, c-means clustering self-organizing maps (SOM) and multidimensional scaling techniques. Advanced examples of the supervised learning models classification and regression trees (CART) and support vector machines [6].

4. Applications of Data Mining in Biomedicine

4.1 Data Mining Models

Data mining applies in descriptive modeling for understanding. In [7], Tseng and Yang use Gene Ontology (GO) to group genes in advance in order to show the potential relations among gene groups and discover the hidden relations between genes set in association with GO terms. It can also be used to predict the outcome of a future observation or to assess the potential risk in a disease situation. Regarding the predictive power, data mining algorithms can learn from past examples in clinical data and model the oftentimes non-linear relationships between the independent and dependent variables, thereby the resulting model representing the formalized knowledge that can often provide a good diagnostic option [8]. Data mining techniques have been widely used to find new patterns and knowledge from biomedical data.

4.2 Recent Development

The typical data mining process involves transferring data originally collected in production systems (such as electronic medical records) into data warehouse, cleaning or scrubbing the data to remove errors and check for format consistency, and then searching the data using statistical model, artificial intelligence (such as neural networks), and other machine learning methods [9]. In [10], Prather et al. employs the KDD for identifying the factors that will improve the quality and cost effectiveness of perinatal care in an extensive clinical database of obstetrical patients. Given the data warehouse of diabetic patients, Breault et al. employ the CART to investigate the factors affecting the occurrence of diabetics [11]. They are surprisingly discovered that younger age predicts bad diabetic control, in which explore a new area to manage the diabetic control in younger age. Similar applications of data mining can also be found in Table 2.

Apart from the diagnostic prediction, the knowledge discovery ability in data mining also demonstrated a good detector in adverse drug events (ADE). In [12], Wilson et al. utilize the KDD techniques in pharmacovigilance for detecting signals earlier than using existing methods. In [13], Lian et al. has pointed out that the prescription is specified by a preference function based on the user's preference in prior clinical experience. Thus, they propose a dose optimization framework based on probability theory. In [14], Susan and Warren have demonstrated that the conditional probability (CP) model is superior in optimizing the drug lists over the multiple linear regression and discriminant analysis models. Concerning the strong relationship between the diagnosis and medication, it formulates a posterior probability (what medication is needed) based on a priori probability (what diagnosis has been made). This approach aligns with the Mediface as purposed by [15].

Table 2. Recent applications of data mining

Author	Description
Megalooikonomou et al. [20]	They introduce statistical methods that aid the discovery of interesting associations and patterns between brain images and other clinical data
Brossette et al. [21]	They design a Data Mining Surveillance System (DMSS) that uses novel data mining techniques to discover unsuspected, useful patterns of nosocomial infections and antimicrobial resistance from the analysis of hospital laboratory data
Antonie et al. [22]	They investigate the use of different data mining techniques for anomaly detection and classification of medical images
Coulter et al. [23]	They examine the relation between antipsychotic drugs and myocarditis and cardiomyopathy
Li et al.[24]	They explore a novel analytic cancer detection method with different feature selection methods and to compare the results obtained on different datasets and that reported by Petricoin et al. in terms of detection performance and selected proteomic patterns
Delen et al.[25]	They use two popular data mining algorithms (artificial neural networks and decision trees) along with a most commonly used statistical method (logistic regression) to develop the prediction models on breast cancer using a large dataset.
Su et al. [26]	They use four different data mining approaches to select the relevant features from the data to predict diabetes
Phillips-Wren et al. [27]	They assess the utilization of healthcare resources by lung cancer patients related to their demographic characteristics, socioeconomic markers, ethnic backgrounds, medical histories, and access to healthcare resources in order to guide medical decision making and public policy

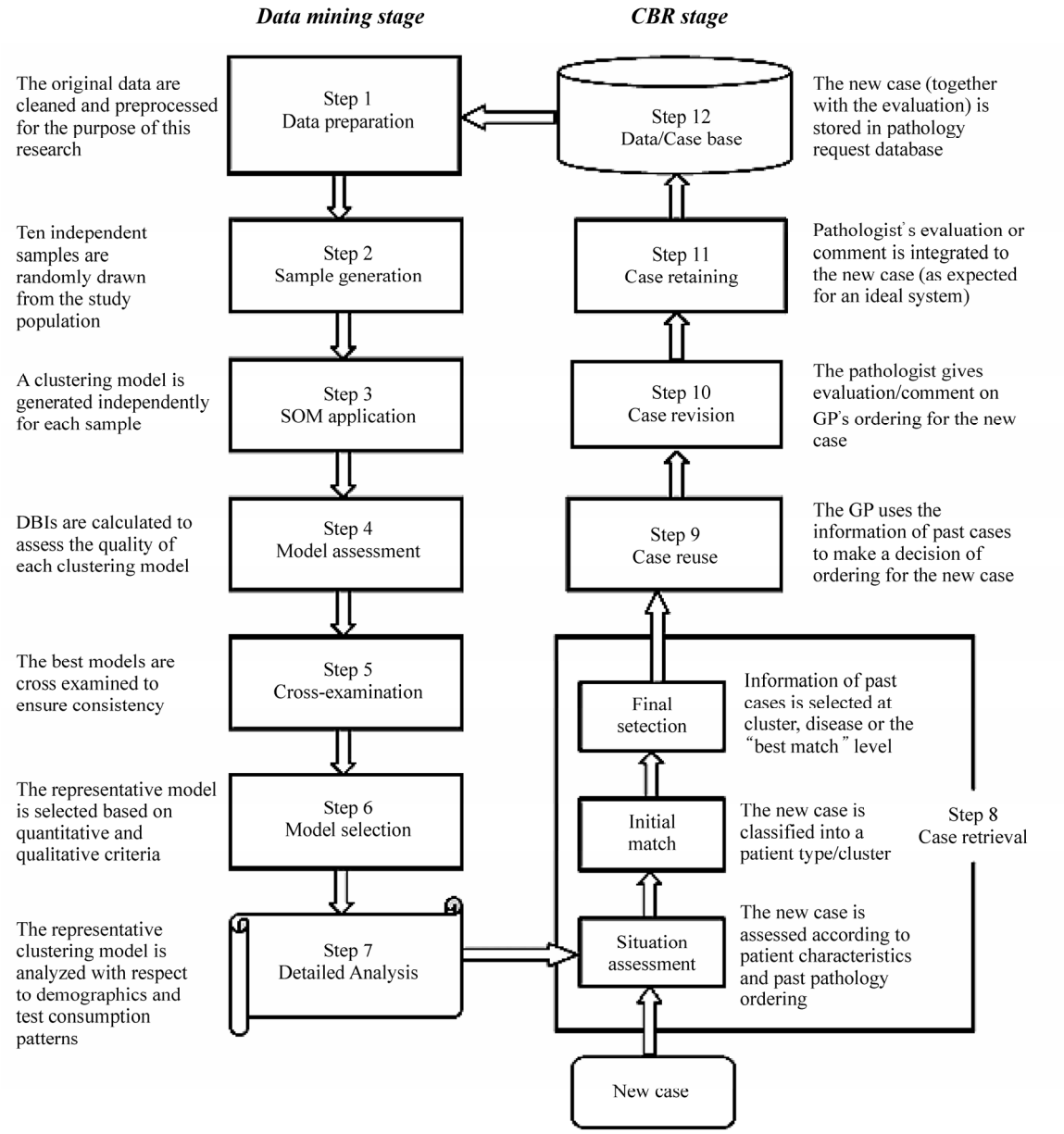


Figure 2. Framework for the integrated approach [17]

In recent years, numerous researchers intend to integrate several data mining and artificial intelligence techniques together to enhance the mining result and support decision making. For example, Kuo et al. integrate the clustering analysis and association rules mining technique to cluster the health insurance database and hence discover the useful rules for each group [16]. In [17], Zhuang et al. combine the data mining and case-based reasoning (CBR) methodologies to provide intelligent decision support for pathology test ordering by GPs. They guarantee the integrated system can enhance the testing ordering in term of evidence based, situational relevance, flexibility and interactivity. In [18], Huang et al. propose a model of a chronic diseases prognosis and diagnosis (CDPD) system by integrating data mining and CBR to support the chronic disease treatment. Compared with traditional coronary artery diseases (CAD) diagnostic methodologies, Tsipouras et al. integrate the decision trees and fuzzy modeling to form a fuzzy rule-based decision support system that obtain a significant improvement compared with artificial neural networks and adaptive neuro-fuzzy inference system [19]. Example of such integration can be found in Figure 2.

All in all, most of the existing data mining applications

are focused on exploring the pattern in sound biomedical databases. With proper structure of the data collected via different medical devices, data mining techniques can serve as a promising tool to convert the information into useful and valuable knowledge to physicians and researchers.

4.3 Current Trend

4.3.1 Multimedia Mining

Classically, databases were formed by tuples of numeric and alphanumeric contents, but with the widespread use of medical information systems, information absorption are now expands to different data types including text, document, image, graphics, speech, audio, hypertext, etc. At the same time, the growth in Internet information can also be considered as a new dimension as a distributed multimedia database of the largest useful information. Concerning the tremendous amount of visual information, it is obvious that the development of data mining techniques in these multimedia data is the next generation in biomedicine. With the widely advanced in digital multimedia technology, numerous researchers introduce several novel data mining techniques, namely image mining, text mining, video mining, and web mining. Below we will discuss these four technology revolution and how does it impact the biomedicine area.

4.3.2 Text Mining

Apart from the medical images and signals, another clinical data that physicians would like to interpret is the unstructured free-text. Regarding there is a lot of information presented in text or document databases, in form of electronic books, research articles, digital libraries, medical dictionaries, etc., several researchers developed a novel data mining approach in extracting useful knowledge from textual data or documents, so called the text mining [28,29]. For example, we can employs text mining techniques to extract the information of protein-protein interaction within three different documents.

In addition to the traditional data mining techniques, text mining uses techniques from many multidisciplinary scientific fields (e.g. text analysis techniques) to gain insight and automatically reveal useful information to the human users. In [30], Cohen and Hunter describe text mining is “the use of automated methods for exploiting the enormous amount of knowledge available in the biomedical literature”. One of the examples of text mining is to manage the health information in Internet and response the needs for those who have health information inquiry in HIV/AIDS [31]. Another common application of text mining is used to extract the information of protein-protein interaction. When given the unstructured text, Zhou et al. employ the semantic parsing and hidden vector state model to mine the knowledge within the text [32]. By setting the annotation PROTEIN_NAME (ACTIVATE(PROTEIN_NAME), the system will automatically generate the result as shown in Figure 3.

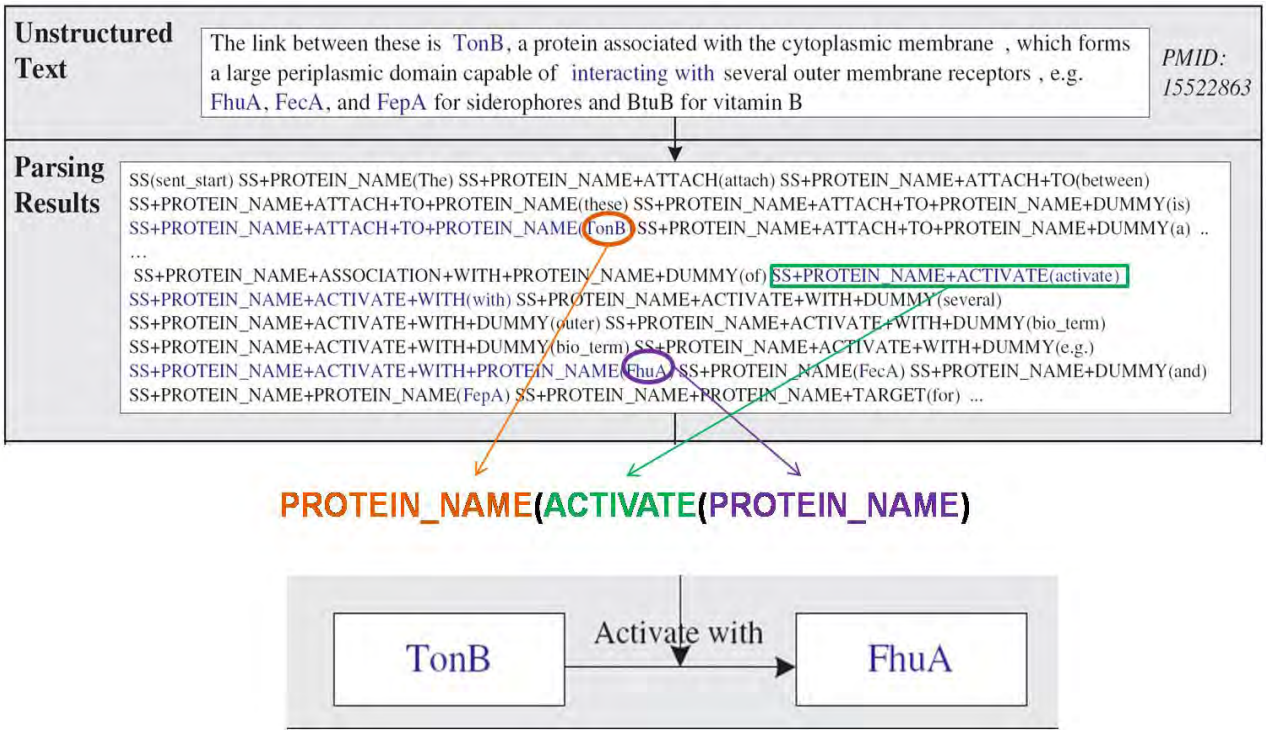


Figure 3. Semantic parsing employed in protein documents [32]

4.3.3 Image Mining

More and more medical procedures employ imaging as a preferred diagnostic tool. Thus, there is a need to develop methods for efficient mining in images databases, which is completely different and more difficult than mining in structured datatypes. Therefore, mining of image data is a challenge problem. Meanwhile, with numerous imaging techniques (such as SPECT, MRI, PET, and collection of ECG or EEF signals) can generate gigabytes of data per day, and heterogeneous nature of image data (like a single cardiac SPECT procedure of one patient may contain dozens of 2D images), image mining has become one of the emerging field in biomedical study. Typically, most of the activities in mining image data are based on the searching, retrieving and comparing of query image with the stored image by its degree of similarity or feature(s). In [22], Antonie et al. present the use of different data mining techniques for tumor classification in digital mammography and they find that associate rule obtains a better result than neural networks. Furthermore, in order to tackle the issue of complicated nature of surrounding of breast tissue, the variation of MCs in shape, orientation, brightness and size, Peng et al. propose knowledge-discovery incorporated genetic algorithm (KD-GA) to search for the bright spots in mammogram and hence evaluate the possibility of a bright spot being a true MC, and adaptively adjust the associated fitness values [34]. Another example, which introduces a notion of image sequence similarity patterns (ISSP) for discovering the possible Space-Occupying Lesion (PSO) in brain images, is presented by [35].

4.3.4 Video Mining

With the advancement in streaming audio and digital TV, more and more video data are stored in which this brings the interest of researchers to discover and explore interesting patterns in the audio-visual content. In order to meet such demand, video mining is developed. In the biomedicine area, it is observed that specialists intend to use cameras to take the video in each operation, which imply there are ample opportunities of applying data mining principles in conjunction with the video retrieval techniques. For example, Zhu et al. introduce a video database management framework and strategies for video content structure and events mining [36]. They first segmented the video shot into groups and hence organized the video shots into a hierarchical structure using clustered scenes, scenes, groups, and shots, in increasing granularity from top to bottom. With a sound structure, audio and video processing techniques are integrated to mine event information, such as dialog, presentation and clinical operation, from the detected scenes.

4.3.5 Web Mining

Internet is growing at a tremendous speed. World Wide Web (WWW) becomes the largest database that ever existed. In particular, many medical literatures are written in electronic format which are widely available and accessible in the Internet nowadays. Therefore, the capability of knowledge discovery and retrieving information from WWW is important to physicians. But, the complexity of web pages and the dynamic nature of data stored in the Internet make adoption of data mining techniques difficult. In [37], web mining is the use of data mining techniques to automatically retrieve, extract and evaluate information for knowledge discovery from the Internet. With its exploratory of hidden information ability, Yu and Jonnalagadda present an approach regarding Semantic Web and mining that can improve the quality of Web mining results and enhance the functions and services and the interoperability of medical information systems and standards in the healthcare field [38].

5. Discussions

Biomedicine has been evolved as a new application area for data mining in recent year. As reflected by the brief literature survey in this study, the current data mining research concentrates on applying the data mining techniques to manage the complex and unstructured data, and in particular in form of visual and textual nature. Although numerous studies resulting satisfactory result of data mining adoption, it is found that data quality is one of the major challenges on impacting the performance in the biomedicine industry. In theory, data mining is a data driven approach as the outcome of data mining heavily depends on the quality and quantity of available data. However, the data in the biomedicine area is rather complex in nature. Thus, in order to enhance the performance of data mining adoption in the domain area, concerns are raised as follow:

a) Huge volume of data

Because of the sheer size of databases, it is unlikely that any of the data mining methods will succeed with raw data. In the field of biomedicine, it is particular true that particular medical experts are required to pre-process the data before adopting data mining. As different medical experts are professional in different medical aspects, therefore it is time consuming and labor intensive to handle the data beforehand.

b) Dynamic nature of data

Databases are constantly updated and adding new information at an alarming rate. For example, new SPECT images (for the same or a new patient), or by replacement of the existing ones (a SPECT had to be repeated because of technical problems). This requires methods that are able to incrementally update the knowledge learned so far.

c) Incomplete or imprecise data

The information collected in a database can be either incomplete or imprecise. To address this problem, fuzzy sets and rough sets were developed explicitly.

d) Noisy data

It is very difficult for any data collection technique to entirely eliminate noise. This implies that data mining methods should be made less sensitive to noise, or care should be taken that the amount of noise in data to be collected in the future will be approximately the same as that in the current data.

e) Missing attribute values

Missing values create a problem for most data mining methods, since nearly all the methods require a fixed dimension for each data object. In fact, this problem is widely encountered in the medical databases because most medical data are collected as a byproduct of patient care activities, rather than for organized research protocols; even in some large medical databases such as breast cancer data set from University of Wisconsin Hospitals, this problem are still existed. Typically, one approach to remedy this problem is to ignore the missing values, or omit any records containing missing values; whereas another approach is to substitute missing values with mostly likely values from obtaining values in the mode or mean, or directly infer missing values from existing values via artificial intelligence method (e.g. case-based reasoning).

f) Redundant, insignificant data, or inconsistent data

The data set may contain redundant, insignificant, or inconsistent data objects and attributes. Generally, medical data can be stored in numeric and textual format; in which a large amount of preprocessing is required in order to make the data useful. For example, misspelled of medical terms is frequently occurred and one medication or condition may be commonly referred to by a variety of names (i.e. stomach and abdominal pain).

In addition to the data quality perspectives, several considerations are also been made:

a) Quality of learning mechanism

Over- and under-learning will affect the performance of data mining in which the learning mechanism will misunderstand the human's preferences and require human to adjust for achieving the goal state.

b) Quality of knowledge representation

Knowledge representation is an important element to represent knowledge in an understandable manner to facilitate the conclusions drawn from knowledge. If the machine is insufficient to store the knowledge discovered, it is also incapable to represent them; thus, such insufficient knowledge will make the machine less intelligent.

c) Nature of problem

When the problem is too complex, chaos, or has not encountered before, the intelligent machine do not have enough knowledge or time to deduce an appropriate result. Using the case of diagnostic decision support as an example, if most of the learning cases and rules are related to some general diagnosis, when there is a new case related to specific diagnosis encountered, the system cannot provide a good solution since there are no rules triggered inside in the system.

As a result, with this study at hand, we can conclude that opportunities to use data mining truly in biomedicine will happen only when the data quality is committed to the level of standard and there are new methods or algorithms to handle the complex data types. Furthermore, adoption of data mining in biomedicine is quite a young field with many issues that still need to be researched and explored in depth. Some further research directions and questions are summarized as follow:

a) An absurd and false model may fit perfectly if the model has enough complexity by comparison to the amount of data available. When the degrees of freedom in parameter selection exceed the information content of the data, this leads to arbitrariness in the final (fitted) model parameters which reduces or destroys the ability of the model to generalize beyond the fitting data. If you've got a learning algorithm in one hand and a dataset in the other hand, to what extent can you decide whether the learning algorithm is in danger of over-fitting or under-fitting? Almost all of the data mining research is done on an ad-hoc base. The techniques are designed for an individual problem. There is no unifying theory.

b) The storage of large multimedia databases is often required to be in compressed form. Data compression if the techniques to reduce the redundancies in data representation. Reducing the storage requirement is equivalent to increasing the capacity of the storage medium. The development of the data compression technology will play a significant role in terms of the performance of data mining. However, it seems the data compression field has so far been neglected by the data mining community.

c) In today's networked society, data care not stored in a single place. Internet has no doubt being the greatest and largest databases that we have ever had. Information inside the internet is often a mixed of text, image, audio, speech, hypertext, graphics and video components. In many cases, databases spread over multiple files in different disks or in different geographical locations. How to handle or collaborate all kind of heterogeneous data in a distributed environment will open up a newer area of development.

d) More and more multimedia data mining systems will be used by medical doctors or clinicians. The design of the system needs to take into consideration of the human perceptual. How to develop a system work synergistically is a subject of ongoing research. In order to achieve the goal, biologist, medical doctors, clinicians and the computing professional all need to work closely together. Any little part missing may lead to the failure of the system design.

6. Conclusions

The well use of the data mining tools in the biomedicine should bring revolutionary impact to the field. The study of biomedical processes is heavily based on the identification of understandable patterns which are present in the data. These patterns may be used for diagnostic or prognostic purpose as well as the analysis of microarrays. Data mining is at the care of the pattern recognition process. Biologist, medical doctors, clinicians and computing professionals should collaborate so that the two fields can contribute to each other. The challenge is for each to widen its focus to attain harmonious and productive collaboration to develop the best practices.

7. Acknowledgement

The authors would like to express their sincere thanks to the Research Committee of The Hong Kong Polytechnic University for financial support of the research work presented in this paper.

REFERENCES

[1] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus, "Knowledge discovery in databases: An overview," AI Magazine, pp. 213–228, 1992.

[2] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, "Data mining: A knowledge discovery approach," Springer, New York, 2007.

[3] J. T. Tou and R. C. Gonzalez, "Pattern recognition principles," Addison-Wesley, London, 1974.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," Wiley, 2001.

[5] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction," Springer, New York, 2001.

[6] J. W. Lee, J. B. Lee, M. Park, and S. H. Song, "An extensive comparison of recent classification tools applied to microarray data," Computational Statistics & Data Analysis, Vol. 48, No. 4, pp. 869–885, 2005.

[7] V. S. Tseng and S. C. Yang, "Mining multi-level association rules from gene expression profiles and gene ontology," in Proceedings IEEE Workshop Life Science Data Mining (held with IEEE ICDM), UK, November 2004.

[8] H. Chen, S. S. Fuller, C. Friedman, and W. Hersh, "Medical informatics–knowledge management and data mining in biomedicine," Springer, 2005.

[9] C. D. Krivda, "Data-Mining Dynamine," Byte, 1995.

[10] J. C. Prather, D. F. Lobach, L. K. Goodwin, J. W. Hales, M. L. Hage, and W. E. Hammond, "Medical data mining: Knowledge discovery in a clinical data warehouse," in Proceedings AMIA Annual Fall Symposium, pp. 101–105, 1997.

[11] J. L. Breault, C. R. Goodall, and P. J. Fos, "Data mining a diabetic data warehouse," Artificial Intelligence in Medicine, Vol. 26, pp. 37–54, 2002.

[12] A. M. Wilson, L. Thabane, and A. Holbrook, "Application of data mining techniques in pharmacovigilance," British Journal of Clinical Pharmacology, Vol. 57, No. 2, pp. 127–134, 2004.

[13] J. Lian, C. Cotrutz, and L. Xing, "Therapeutic treatment plan optimization with probability density-based dose prescription," Medical Physics, Vol. 30, No. 4, pp. 655–666, 2003.

[14] E. G. Susan and J. M. Warren, "Statistical modelling of general practice medicine for computer assisted data entry in electronic medical record systems," International Journal of Medical Informatics, Vol. 57, No. 2-3, pp. 77–89, 2000.

[15] J. R. Warren, A. Davidovic, S. Spenceley, and P. Bolton, "Mediface: Anticipative data entry interface for general practitioners," in Proceedings Computer Human Interaction Conference 1998, pp. 192–199, 1998.

[16] R. J. Kuo, S. Y. Lin, and C. W. Shih, "Mining association rules through integration of clustering analysis and ant colony system for health insurance database in Taiwan," Expert Systems with Applications, Vol. 33, pp. 794–808, 2007.

[17] Z. Y. Zhuang, L. Churilov, F. Burstein, and K. Sikaris, "Combining data mining and case-based reasoning for intelligent decision support for pathology ordering by general practitioners," European Journal of Operational Research, Vol. 195, No. 3, pp. 662–675, 2009.

[18] M. J. Huang, M. Y. Chen, and S. C. Lee, "Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis," Expert Systems with Applications, Vol. 32, No. 3, pp. 856–867, 2007.

[19] M. G. Tsipouras, T. P. Exarchos, D. I. Fotiadis, A. P. Kotsia, K. V. Vakalis, K. K. Naka, and L. K. Michalis, "Automated diagnosis of coronary artery disease based on data mining and fuzzy modeling," IEEE Transactions on Information Technology in Biomedicine, Vol. 12, No. 4, pp. 447–457, 2008.

[20] V. Megalooikonomou, J. Ford, L. Shen, F. Makedon, and A. Saykin, "Data mining in brain imaging," Statistical Methods in Medical Research, Vol. 9, No. 4, pp. 359–394, 2000.

[21] S. E. Brossette, A. P. Sprague, W. T. Jones, and S. A. Moser, "A data mining system for infection control surveillance," *Methods of Information in Medicine*, Vol. 39, No. 4-5, pp. 303–310, 2000.

[22] M. L. Antonie, O. R. Zaiane, and A. Coman, "Application of data mining techniques for medical image classification," in Proceedings Second International Workshop on Multimedia Data Mining, pp. 94–101, 2001.

[23] D. M. Coulter, A. Bate, R. H. B. Meyboom, M. Lindquist, and R. Edwards, "Antipsychotic drugs and heart muscle disorder in international pharmacovigilance: Data mining study," British Medical Journal, Vol. 322, pp. 1207–1209, 2001.

[24] L. Li, H. Tang, Z. Wu, J. Gong, M. Gruidl, J. Zou, M. Tockman, and R. Clark, "Data mining techniques for cancer detection using serum proteomic profiling," Arti-

ficial Intelligence in Medicine, Vol. 32, No. 2, pp. 71–83, 2004.

[25] D. Delen, G. Walker, and A. Kadam, "Predicting breast cancer survivability: A comparison of three data mining methods," Artificial Intelligence in Medicine, Vol. 34, No. 2, pp. 113–27, 2005.

[26] C. T. Su, C. H. Yang, K. H. Hsu, and W. K. Chiu, "Data mining for the diagnosis of type II diabetes from three-dimensional body surface anthropometrical scanning data," Computers & Mathematics with Applications, Vol. 51, No. 6–7, pp. 1075–1092, 2006.

[27] G. Philips-Wren, P. Sharkey, and S. Morss, "Mining lung cancer patient data to assess healthcare resource utilization," Expert Systems with Applications: An International Journal, Vol. 35, No. 4, pp. 1611–1619, 2008.

[28] M. Hearst, "Untangling text data mining," in the Proceedings ACL'99: The 37th annual meeting of the association for computational linguistics, University of Maryland, June 1999.

[29] H. Chen, "Knowledge management systems: A text mining perspective," Tucson, AZ, The University of Arizona, 2001.

[30] K. B. Cohen and L. Hunter, "Getting started in text mining," PLoS Computational Biology, Vol. 4, No. 1, doi: 10.1371/journal.pcbi.0040020, 2008.

[31] Y. Ku, C. Chiu, B. H. Liou, J. H. Liou, and J. Y. Wu, "Applying text mining to assist people who inquire HIV/AIDS information from Internet," in Proceedings ISI 2008 Workshops, pp. 440–448, 2008.

[32] D. Zhou, Y. He, and C. K. Kwoh, "Validating text mining results on protein-protein interactions using gene expression profiles," in Proceedings International Conference on Biomedical and Pharmaceutical Engineering 2006, pp. 580–585, 2006.

[33] Y. Peng, B. Yao, and J. Jiang, "Knowledge-discovery incorporated evolutionary search for microcalcification detection in breast cancer diagnosis," Artificial Intelligence in Medicine, Vol. 37, No. 1, pp. 43–53, 2006.

[34] H. Pan, Q. Han, X. Xie, Z. Wei, and J. Li, "A Similarity retrieval method in brain image sequence database," Advanced Data Mining and Applications, Vol. 4632, pp. 352–364, 2007.

[35] X. Zhu, W. G. Aref, J. Fan, A.C. Catlin, and A. K. Elmagarmid, "Medical video mining for efficient database indexing, management and access," in Proceedings 19th International Conference on Data Engineering, pp. 569–580, 2003.

[36] R. Kohavi, B. Masand, M. Spilipoulou, and J. Srivastava, "Web mining," Data Mining and Knowledge Discovery, Vol. 6, pp. 5–8, 2002.

[37] W. D. Yu and S. R. Jonnalagadda, "Semantic web and mining in healthcare," in Proceedings 8th International Conference on e-Health Networking, Applications and Services, pp. 198–201, 2006.

[38] S. Mitra and T. Acharya, "Data mining: Multimedia, soft computing and bioinformatics," John Wiley & Sons, Inc., New Jersey, 2003.

J. Software Engineering & Applications, 2009, 2: 160-164
doi:10.4236/jsea.2009.23023 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

A Solution Based on Modeling and Code Generation for Embedded Control System

Guohua WU, Dongwu CHENG, Zhen ZHANG

School of Computer Science, Hangzhou Dianzi University, Hangzhou, China.
Email: wugh@hdu.edu.cn, cdw8411@163.com, zwhnz@zj165.com

Received April 2nd, 2009; revised May 30th, 2009; accepted July 4th, 2009.

ABSTRACT

With the development of computer technology, embedded control system plays an important role in modern industry. For the embedded system, traditional development methods are time-consuming and system is not easy to maintain. Domain-specific modeling provides a solution for the problems. In this paper, we proposed development architecture for embedded control systems based on MIC. GME is used to construct meta-model and application model, model interpreter interprets model and stores model information in xml format document. The final cross-platform codes are automatically generated by different templates and xml format document. This development method can reduce time and cost in the lifecycle of system development.

Keywords*: *Domain-Specific Modeling, Model Interpreter, Code Generation, Embedded Control System

1. Introduction

The processing capability of generous purpose microprocessor is increasing and moving system development emphasis from hardware to software. In order to meet the embedded system requirements, software development process becomes sophisticated. Developing embedded control system with safety-critical and real-time characteristics by the traditional method is time-consuming.

Matlab/Simulink [1] focuses on data visualization, algorithms, analysis and numeric computing. The code can be automatically generated from model. However, it is not adequate for developing embedded control system. Giotto [2] is a time-triggered language for embedded control system which is developed by university of Berkeley. It supports the automation of control system designed by strictly separating platform-dependent functionality from scheduling and communication. However, developer must develop different virtual machines for application, because application is interpreted on two virtual machines: the scheduling and embedded machine. In addition, the code generated by Giotto complier is timing-code. Model-based development method is from high level abstraction to build application. Currently, unified modeling language (UML) [3] is the most popular modeling language. Although some diagrams are suitable for automatic code generation, the implementation must be done by hand. As a general purpose modeling language, UML is unable to describe embedded control system characteristics such as deadline, and fault-tolerance.

MIC [4–6] is a modeling framework based on *models* and *generation.* It employs domain *models* to represent system, its environment, and the relationships between them. We use GME [7] for constructing meta-model and application model. GME is a configurable toolkit and supports domain-specific modeling. In the modeling process, UML and Object Constraint Language (OCL) [8] are used to express meta-model and constraints. The BON2 [9] component is used to interpret *model.* In this paper, we proposed an embedded control system development framework based on domain-specific modeling. The framework contains four layers: meta-meta model layer, meta-model layer, model layer and implementation layer. In the model interpreting process, we interpret model and store information in xml format document. The final code is generated by templates files and xml document, implementing cross-platform codes automatic generation.

The paper is organized as follows. In section 2, we illustrate the embedded control system development architecture. In section 3, we illustrate a meta-modeling process and take an example of state machine in detail. In section 4, we describe a method for model interpretation and code generation. In section 5, we make a conclusion.

2. Embedded Control System Developments Architecture

According to the characteristics of the embedded control system and model-integrated computing (MIC), we divided model architecture into four layers (see Figure 1).

1) Meta-meta model layer. Define meta-model modeling language, which is a general-purpose language and independent on domain.

2) Meta model layer. This is the core and infrastructure of implementing domain model.

3) Model layer. Domain developers construct application model according to domain knowledge and relevant rules.

4) Implementation layer. This is the concrete implementation of application model. According to tasks distribution in the model, execute software and hardware relevant operations.

3. Construct Meta-Model for Embedded Control System

Meta-model is expressed by UML class diagrams and OCL expressions, which specifies the static semantic and syntax of domain-specific language. In the process of meta-model modeling, the characteristics that impact system function and performance such as currency, deadline and the worst case execution time should be specified.

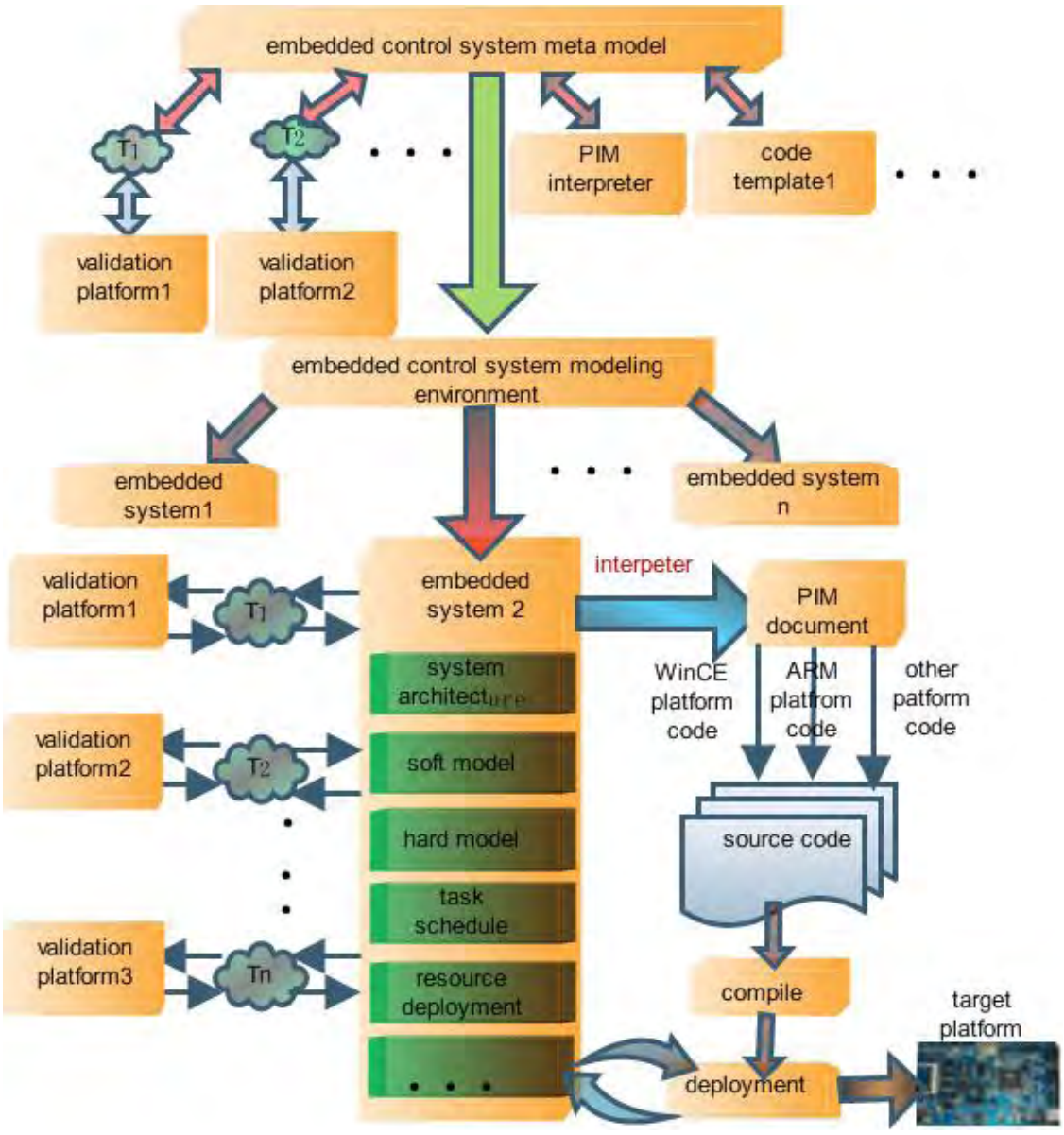


Figure 1. Embedded control system model architecture

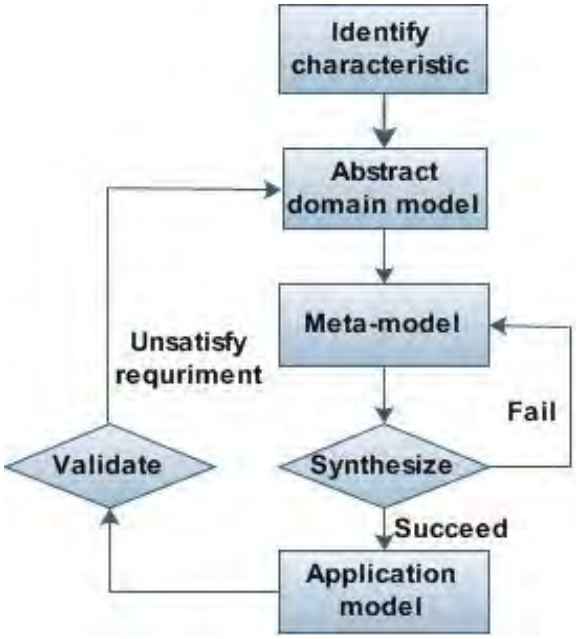


Figure 2. The meta-model modeling process

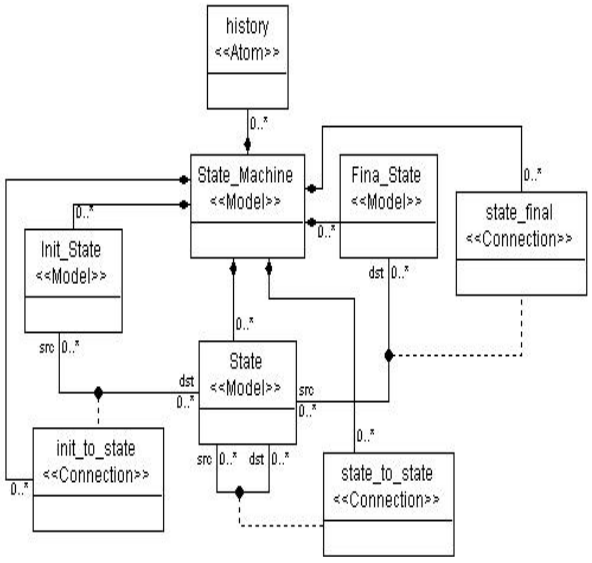


Figure 3. State machines meta-mode

3.1 Meta-Model Modeling Processes

Formalization of modeling language to be the corresponding meta-model is a recursive process (see Figure 2).

1) Identifying the characteristics and properties of the embedded control system such as real-time, safety-critical and concurrent.

2) According to the abstract principles, acquiring suitable model from characteristic and properties.

3) Acquiring a higher level meta-model by formalizing model which is used to construct domain models.

4) Synthesizing domain application model by domain modeling language. If application model can not be correct constructed, retry to acquire meta-model until it is successful.

5) Validating the application model against system requirement. If the application model can not meet the system requirements, retry to abstract domain model again.

3.2 Meta-Model of FSM

State machine is an important component in the embedded control system. The state machine model (see Figure 3), including initial state, terminal state, and state transition. Event triggers state transition and each state has sequential behaviors. Initial state and terminal state denote the beginning and ending of the state machines respectively. State transition is a common behavior and each state may consist of multiple transitions.

The meta-model in form of class diagram together with the constraints expressed in OCL provides a complete formal definition for model.

3.3 Embedded Control System Modeling Language

Domain specific modeling language (DSML) employs domain-specific concept symbols to specify restrict yet precise semantics. Formally, a modeling language is a five-tuple of concrete syntax (C), abstract syntax (A), semantic domain (S), semantic mapping (Mc) and syntactic mapping (Ms) [10].

$$L=<C, A, S, Mc, Ms>$$

1) Abstract syntax (A), defining the concepts, relationships, and integrity constraints available in the language.

2) Concrete syntax (C), defining the specific (graphical or textual) notation that is used to express models. It may be graphical, textual or mixed.

3) Semantic domain (S), which is formal semantic defined by mathematical formalism in terms of the meaning of the model is explained.

4) Semantic mapping (Ms), $A \rightarrow S$ mapping relates syntactic concepts to the semantic domain.

5) Syntactic mapping (Mc), $A \rightarrow C$ mapping syntactic constructors (graphical, textual, or both) to the elements of abstract syntax.

The primary participants in domain-specific modeling are modelers. In the process of embedded control system development, domain specific modeling language with precise syntax and semantic will be defined from different aspects.

4. Model Interpretation and Code Generation

Model interpreter plays an important role in the embed-

ded control system development. We can develop different interpreters for meeting requirements. The interpreter is similar to a complier of advanced programming language. We take BON2 component to interpret model which is provided by GME. BON2 consists of class and interface, traversing objects in the model. The final cross-platform codes are automatic generated by different templates.

4.1 Model Interpretation

Model interpreter is a component of GME, which is used to acquire objects' information in application model. Developers can build various interpreters according requirements. The interpreters interpret model and store model information in PIM document (see Figure 4), facilitating data transition and access.

```
<?xml version="1.0"encoding="utf-8"?>
<root>
    <state>
        <name>Ready</name>
    </state>
    <state>
        <name>Run</name>
    </state>
    <state>
        <name>Stop</name>
    </state>
    <transition>
        <src>Ready</src>
        <dest>Run</dest>
    </transition>
    <transition>
        <src>Run</src>
        <dest>Stop</dest>
    </transition>
    <transition>
        <src>Stop</src>
        <dest>Ready</dest>
    </transition>
</root>
```

Figure 4. Model Interpretation PIM document

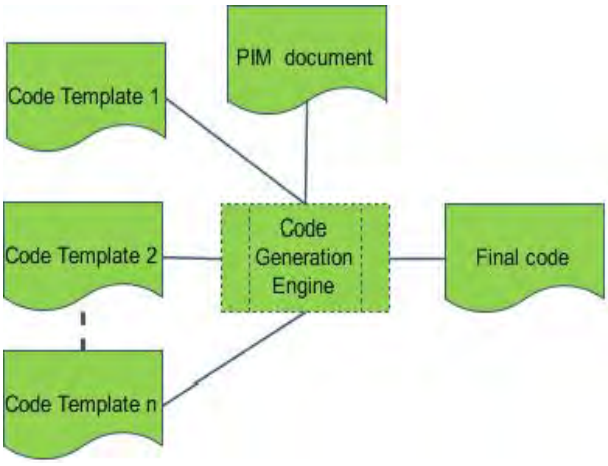


Figure 5. Code generation architecture

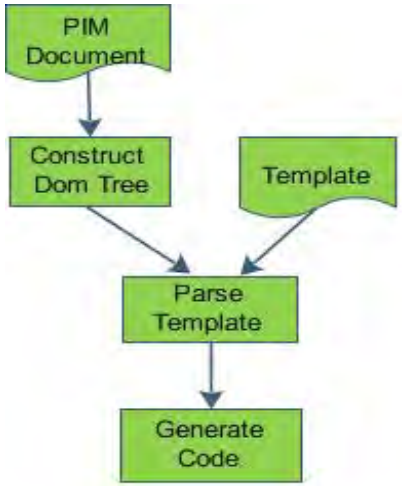


Figure 6. Process of code generation

4.2 Based on Template Code Generation

In the process of code generation, we use different templates to meet the needs of different platforms, which is similar to macros [11]. If there is no available template for specific application, the application developer can develop new templates according to the requirements. In this way, we can guarantee that the code supports different platforms (Figure 5).

The workflow of code generation engine consists of two steps (see Figure 6). Firstly, invoking xml DOM parser to parse PIM and building DOM tree. Secondly, parsing template contents and replacing template contents with DOM tree.

5. Conclusions

In this paper, we present a solution for developing the embedded control system. Relevant meta-model and model are expressed by class diagrams and a set of OCL constraints. A model interpreter is developed and the model information is interpreted, which is stored in PIM document. According to the system requirements, if we need new functions and want to support cross-platform, we have to construct new templates. This solution speeds up the application development and reduces the cost. What's more, the application is easy to maintain by the modified meta-model.

REFERENCES

[1] P. Barnard, "Software development principles applied to graphical model development," In AIAA Modeling and simulation Technologies conference an Exhibit, San Francisco, August 2005.

[2] T. A. Henzinger and C. M. Kirsch, "The embedded machine: Predictable, portable real-time code," Proceedings of the International conference on Programming Lan-

guage Design and Implementation (PLDI), ACM press, pp. 315–326, 2002.

[3] Object Management Group, "OMG unified modeling language specification," http://www.uml.org/, 2007.

[4] J. Sztipanovits and G. Karsai, "Model-integrated computing," IEEE computer, pp. 110–112, April, 1997.

[5] G. Karsai, A. Agrawal and A. Ledeczi, "A meta–model -driven MDA process and its tool," Workshop in software Model Engineering, 2003.

[6] G. Karsai, J. Sztipaovits, A. Ledeczi and T. Bapty. "Model-integrated development of embedded software," In processing of the IEEE, pp.145–164, 2003.

[7] A. Ledeczi, M. Maroti, G. Karsai, J. Garrett, J. Sprinkle, et al., "The generic modeling environment," Workshop on Intelligent Signal Processing Budapest, Hungary, May 17, 2001.

[8] Object Management Group, "Object constraint language," http://www.omg.org/docs/ptc/03-10-14.pdf, 2003.

[9] General Modeling Environment, http://www.isis.vanderbilt.edu/sites/default/files/GMEUMan.pdf, 2005.

[10] T. Clark, A. Evans, S. Kent and P. Sammut, "The MMF approach to engineering object-oriented design language," Workshop on Language Description, Tools and Applications, April 2001.

[11] C. Buckl, A. Knoll, and G. Schrott. "Model-based development of fault-tolerant embedded software," in second International symposium on Leveraging Applications of Formal Method, Verification and Validation, pp. 113–120, 2006.

J. Software Engineering & Applications, 2009, 2: 165-172
doi:10.4236/jsea.2009.23024 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Product Maintainability Design Method and Support Tool Based on Feature Model

Yufeng DING

School of Mechanical and Electrical Engineering, Wuhan University of Technology, Wuhan, China.
Email: dingyf@whut.edu.cn

Received June 5th, 2009; revised June 29th, 2009; accepted July 7th, 2009.

ABSTRACT

Maintainability is an important character which is given by product design process. The maintainability design criteria and measure index used in product maintainability analysis are summarized and discussed in this paper. A product maintainability design method is studied by integrating the product feature model, maintainability design criteria with measure index. Product feature model can be built on the basis of the product feature library quickly. Product feature library for steam turbine design is created by using SolidWorks design library origination structure. A methodology which supports the design and development of product maintainability design support tool (PMDSTs) is put forward. The function of PMDSTs is designed by using UML (Unified Modeling Language) use case diagram, it is developed by using VC++ 6.0. The maintainability analysis application case of steam turbine-generator system is given at last.

Keywords*: *Maintainability Design, Product Design Process, Steam Turbine, Solidworks

1. Introduction

If a product has poor maintainability, the maintenance activities which have to be performed on it during its life cycle are difficult, it will result in increasing the costs, and also more time is required to accomplish the maintenance tasks. Designs for maintainability had played an important role in the complex product deign process. Maintainability is the probability that required maintenance will be successfully completed in a given time period. It is a design characteristic that affects accuracy, ease, and time requirements of maintenance actions. It may be measured by combining factors such as frequency of maintenance, maintenance costs, elapsed maintenance or repair times, and labor hours etc.

Design for maintainability requires a product that is serviceable and supportable—better yet if the design includes a durability feature called reliability (absence of failures) then you can have the best of all worlds [1].

Maintainability is an important character which is given by product design, it makes easy to be repaired for the mechanical system. It has a specific effect on the maintenance cost of mechanical systems [2].

In order to realize product design for maintainability, some approaches and software tools have been studied and developed. A maintainability evaluation approach based on fuzzy logic is presented in [3], fuzzy linguistic variables are employed in order to represent and handle the design data available early in the design process. The measure tool of the product maintainability is developed. Maintainability and safety indicators at design stage for mechanical products are studied in [4]. The assessment procedure uses product 3D (Dimension) CAD (Computer Aided Design) model and associated semantic matrix gathering information from the product components' criticality and reliability.

2. Product Integration Maintenance Model

Product CAD model doesn't include product maintenance feature information; it can not provide support for product maintenance. So it needs to build product integration model by combining CAD model and maintenance feature information. Product integration model is composed of shape feature F_s, heat treatment feature F_h, management feature F_m and maintenance feature F_{ma}. They can be united altogether to build a whole product feature model. The product feature classification is shown in the Figure 1. Maintenance feature is used for describing maintainability design and relevant information. It includes maintenance organization feature, maintainability feature, maintenance resource, maintenance process, fault feature, using demand, maintenance program, maintainability qualitative demand, quantitative demand, maintainability design analysis result, person model info etc. Maintainability program includes objectives, organization, maintainability design criteria, poli-

cies and procedures, organizational interfaces, program tasks, evaluation and subcontractor/supplier activity etc. [5].

Product maintenance integration model is shown in Figure 2. C_i means ith component. It can be a part or a sib-assembly. R(i,j) represents the assembly relation between ith component and jth component. It is marked in the link arc between ith component and jth component. Every component has an attribute table, all attribute tables are saved in the attribute table. Every attribute table file saves all feature informations of component C_i. Some data files such as design parameter table (excel file), design intent (word file), maintenance plan (Microsoft project file) are stored in design document data or maintenance data. They have a link in the Tag file. Tag file will look for the relevant file by the link [5].

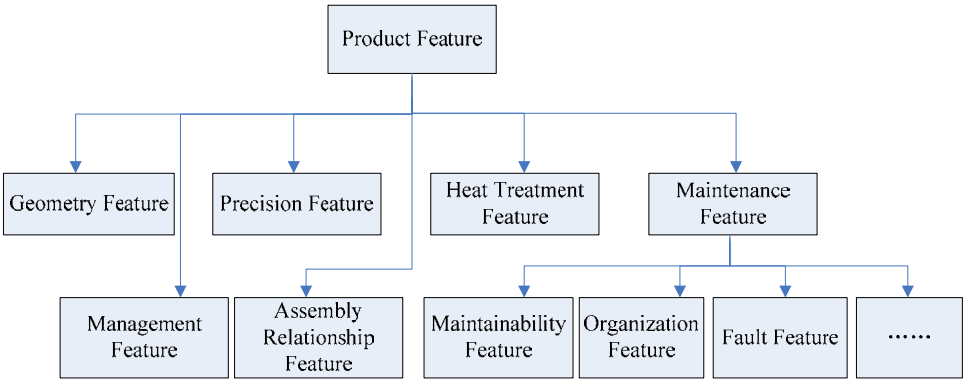


Figure 1. Product feature classification

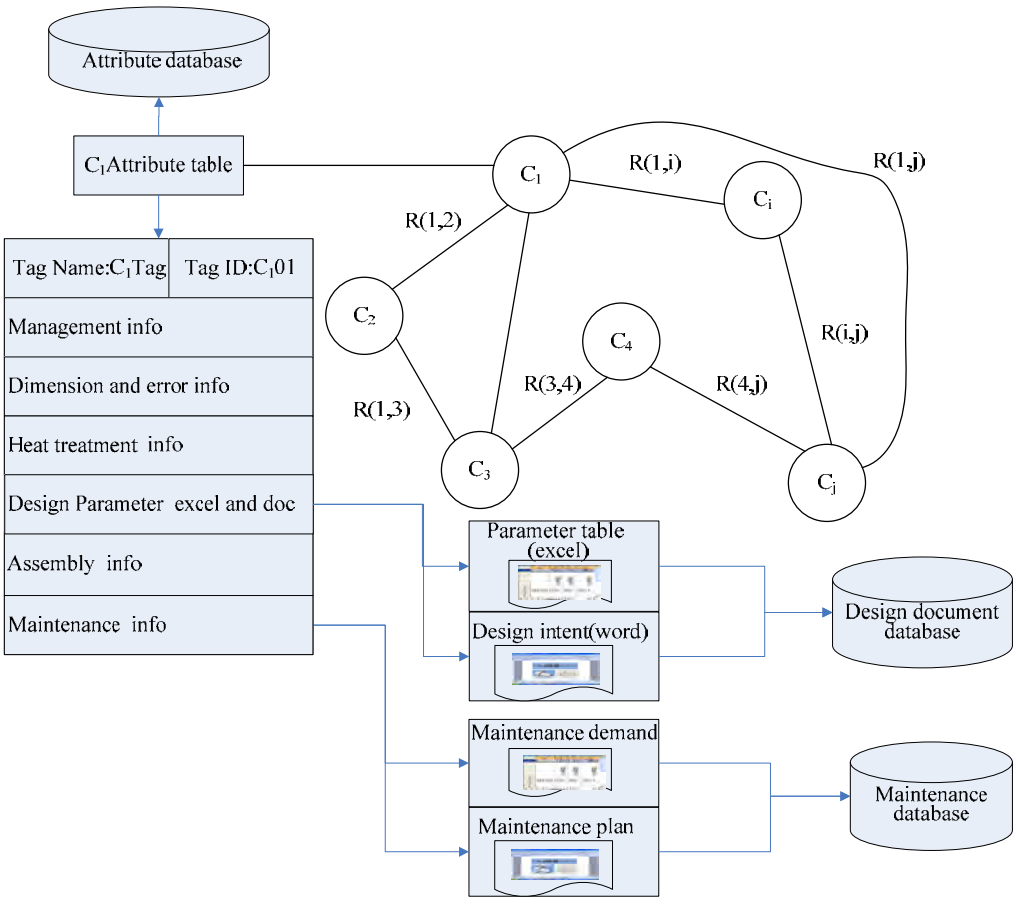


Figure 2. Product maintenance integration model

3. Design for Maintainability

3.1 The Principles of Design for Maintainability

The design engineer is anxious to verify if product maintenance tasks can be accomplished with the available maintenance logistics and evaluate the performance of these tasks in design stage. The designer must change his/her design solution if the maintenance tasks are difficult, or even impossible to be accomplished in the given conditions. Some of the important general design guidelines that maintainability professionals have developed are shown in Figure 3 [6].

Maintainability criteria are composed of intrinsic criteria and contextual criteria [4]. Intrinsic criteria includes repairability, ability to be repaired after failure or damage; accessibility, easiness to reach a component inside the assembly; assemblability, ability to be assembled from an assembly; disassemblability, ability to be removed from an assembly; standardization, standard component or equipment; interchangeability, ability to be replaced with another component; survivability, ability of the product to continue to work after the failure of a considered component. Contextual criteria includes redundancy, for components existing in multiple equivalent occurrences; Competencies, human required to diagnose and to repair; Toolings, maintenance equipments like keys, screwdrivers...; Logistics, delivery of spare parts, transportation of maintenance team...; Environment, working conditions like lighting, temperature...; Delectability, easiness to detect failure and components concerned with; Testability, ability for a component or a sub-system to be tested...; Maneuverability, ability for a component or sub-system to be handled; Auto diagnostic: ability for a system to perform self-testing procedures.

3.2 Maintainability Measure

Maintainability, as a characteristic of design, can be defined on the basis of a combination of the following factors. They are Maintenance times, Maintenance frequency and Maintenance cost. The three factors are dependant on the fact that the system is operated and maintained in accordance with prescribed procedures and resources. From a systematic perspective, maintenance includes corrective maintenance and preventive maintenance. And from a software perspective, maintenance includes adaptive maintenance and perfective maintenance.

There are several approaches to evaluate the maintainability of a product at the design stage. They are maintainability design checklists, maintainability evaluation using physical mock-ups, maintainability evaluation using digital mock-ups and virtual reality and maintainability evaluation using quantitative approaches.

The measures used in maintainability analysis include mean time to repair, mean active preventive maintenance time, and mean active corrective maintenance time, maximum corrective maintenance time, and mean maintenance downtime [7].

3.2.1 Mean Time to Repair (MTTR)

Maintenance time to repair (MTTR) is very crucial and it depends mainly on the product configurations. MTTR measures the elapsed time required to perform a given maintenance activity and is subsequently used to calculate system availability and downtime. Exponential, log-normal, and normal probability distributions can all represent mean time to repair. The normal distribution is normally assumed for mechanical or electromechanical equipment with a remove-and-replace maintenance concept.

$$T_{mttr} = (\sum_{i}^{m} \lambda_i T_i) / \sum_{i}^{m} \lambda_i \quad (1)$$

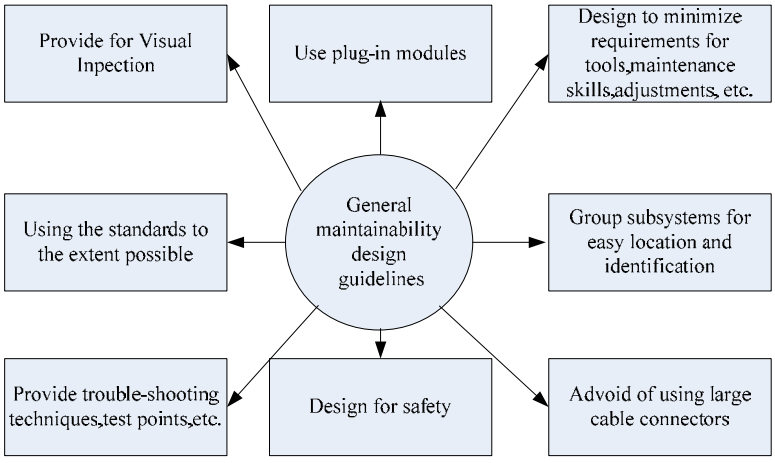


Figure 3. General maintainability design guidelines

3.2.2 Mean Preventive Maintenance Time

Preventive maintenance activities such as inspections, calibrations, and tuning keep equipment at a specified performance level. The objective of a preventive maintenance program is to postpone the point at which the equipment or any of its components wears out or breaks down.

$$T_{mp} = \frac{\sum_{i}^{k} (T_{mpi})(F_{pti})}{\sum_{i}^{k} F_{pti}} \quad (2)$$

T_{mp} is the mean preventive time. T_{mpi} is the elapsed time for preventive maintenance task i, for i= 1,2,3 . . . . . k.F_{pti} is the frequency of preventive maintenance task i, for i =1, 2, 3 , . . . ,k. k is the number of preventive maintenance tasks.

3.2.3 Median Corrective Maintenance Time

Mean corrective time is a composite value representing the arithmetic average of the individual maintenance cycle times. Maintainability is the ability of a product to be maintained. Calculation of the median corrective maintenance time depends on the distribution describing time to repair. The median corrective maintenance time for lognormal distributed repair time is given by

$$T_{med} = T_{mttr} / \exp(\sigma^2) \quad (3)$$

σ^2 is the variance around the mean value of the natural logarithm of repair time.

3.2.4 Maximum Corrective Maintenance Time

This measures the time required to complete all potential repair activities up to a given percentage, often the 90th or 95th percentiles. The maximum corrective maintenance time for lognormal distributed is

$$T_{mcm} = \text{antilog}(T_m + k\sigma) \quad (4)$$

T_{mcm} is the maximum corrective maintenance time. Tm is the mean of the logarithms of the repair times. σ is the standard deviation of the logarithms of repair times. k is equal to 1.28 or 1.65 for the 90th and 95th percentiles.

3.2.5 Mean Maintenance Downtime

This is the total time needed either to restore equipment to a specified performance level or to maintain it at that level of performance. Thus it includes active corrective and preventive maintenance times, administrative and logistic delay times.

$$T_{mmd} = T_{mam} + T_{ad} + T_{ld} \quad (5)$$

T_{mmd} is the mean maintenance downtime. T_{mam} is the mean active maintenance time, or mean time required to conduct corrective and preventive maintenance related tasks. T_{ad} is the administrative delay time. T_{ld} is the logistic delay time.

3.2.6 Maintenance Function

The maintainability functions are used to predict the probability that a repair, beginning at time t = 0, will be accomplished in a time t. The maintainability function, m(t) for any distribution is expressed by

$$m_t = \int_0^t f_r(t)dt \quad (6)$$

t is time.$f_r(t)$ is the probability density function of the repair time.

3.2.7 Maintenance Accessibility Evaluation

Accessibility is the relative ease with which a part or piece of equipment can be reached for service, replacement, or repair. The lack of accessibility is an important maintainability problem and a frequent cause of ineffective maintenance.

The evaluation may also be performed by assigning to each maintainability criterion, a numerical value between 0 and 1 using a table, like the one listed in Table 1 [3].

3.3 Maintainability Cost Analysis

Maintainability is an important factor in the total cost of equipment. An increase in maintainability can lead to reduction in operation and support costs. For example, a more maintainable product lowers maintenance time and operating costs. Furthermore, more efficient maintenance means a faster return to operation or service, decreasing downtime.

The data to be input into a life cycle cost model include the purchase price of the product, mean time between failures (MTBF), MTTR, average material cost of a failure, labor cost per preventive maintenance action, labor cost per corrective maintenance action, installation costs, training costs, the warranty coverage period cost of carrying spares in inventory, and shipment forecasts over the course of the product's useful life [8].Corrective Maintenance cost estimation model estimates the corrective maintenance labor cost for a piece of equipment. The annual cost is expressed by

Table 1. Accessibility evaluation

Accessibility	Value
All the parts are directly accessible and placed in the same area	1
All the parts are directly accessible and placed in different areas	0.8
Some parts are not directly accessible, but those parts are maintenance free	0.6
Some parts are accessible after disassembling a fast disassembling entity (a screw, etc.)	0.4
The majority of the parts is accessible by disassembling one or more entities	0

$$C_{CM} = \frac{(SOH)(C_L)(T_{mttr})}{T_{mtbf}} \quad (7)$$

SOH represents the scheduled operating hours of the equipment. C_L is the maintenance labor cost per hour. T_{mtbf} is the mean time between failures for the equipment. T_{mttr} is the mean time to repair for the equipment.

3.4 System Reliability Analysis

Reliability can be defined as the probability that a system will perform properly for a specified period of time under a given set of operating conditions. Implied in this definition is a clear-cut criterion for failure.

Let R_i be the reliability of subsystem i and r_{ij} be the reliability of component, in subsystem j,

$1 \le j \le n_i$ i=1,2, …,k. Then

$$R_i(t) = \prod_{j=1}^{n_i} r_{ij}(t) \quad (8)$$

The system reliability, say R(t), is given by

$$R_i(t) = 1 - \prod_{i=1}^{k} \left(1 - \prod_{j=1}^{n_i} r_{ij}(t)\right) \quad (9)$$

The system consists of k subsystems connected in parallel, with subsystem i consisting of n_i components in series for i=1,2,y,k. Such a system is called a series–parallel system [9]. Figure 4 shows the diagram of a series–parallel system [10].

The system mean time to failure (MTTF) can be derived in the following form:

$$T_{mttf} = \frac{1}{\lambda} \sum_{l=1}^{k} \sum_{1 \le i_1 < \ldots \le k} \frac{(-1)^{l+1}}{n_i^{(l)}} \quad (10)$$

4. The Development and Application of Product Maintainability Design Support Software

4.1 Product Feature Library

A library feature is a frequently used feature, or combination of features. Many features can be created once and then save in a library for future use. Product feature is organized according to product feature classification of Figure 1.The product feature can be built and saved in design library of Solidworks. The directory structure can be built in the directory of install_directory\data\design library. To create a library feature that includes references, it is needed to dimension the library feature relative to the base part on which designer create it. References create dimensions used to position the library feature (*.sldlfp) on the model (*.sldprt). Steam turbine feature library in the SolidWorks is shown in Figure 5.

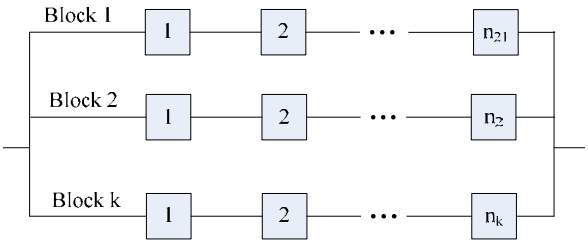


Figure 4. Series–parallel system

4.2 Product Feature Modelling Based on Feature Library

Industry steam turbine is composed of about tens of thousands of parts. The rotor, blade and cylinder of steam turbine work under the condition of high temperature and impulsion. They have high manufacture precision. The rotor of steam turbine is the most important, highest precision and most complicated part in the steam turbine product. Rotor includes thousands of dimensions.

For example, steam turbine rotor axis can be divided into five parts, they are front axis feature(A_{FA}), front seal feature(A_{FGS}), whole blade wheel feature(A_{BW}), back seal feature(A_{BGS}) and back axis feature(A_{BA}) .It can be represented in equation (11).

$$Fr = A_{FA} \cup A_{FGA} \cup A_{CA} \cup A_{BGA} \cup A_{BA} \quad (11)$$

All features of steam turbine rotor can be organized in the design feature library in the SolidWorks. But the first step must be done is to classify the features according to component classification and recognize the feature dimension. Some dimension relation can be built in some equation. The maintenance feature information had been organized into attribute table which is integrated with the geometry feature. Steam turbine feature library in the SolidWorks is shown in Figure 5.

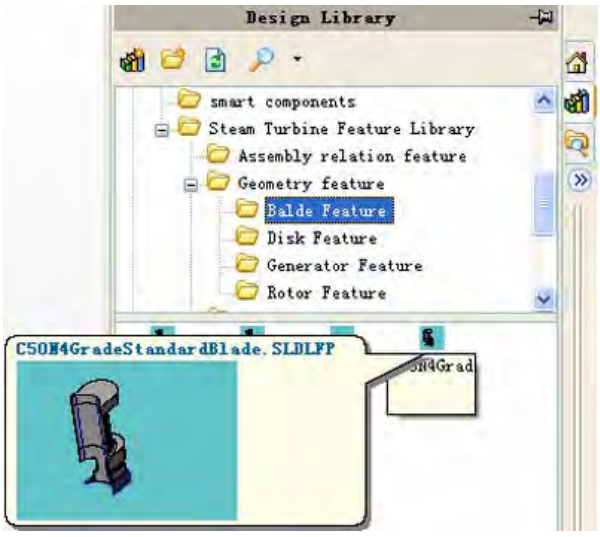


Figure 5. Steam turbine feature library in the SolidWorks

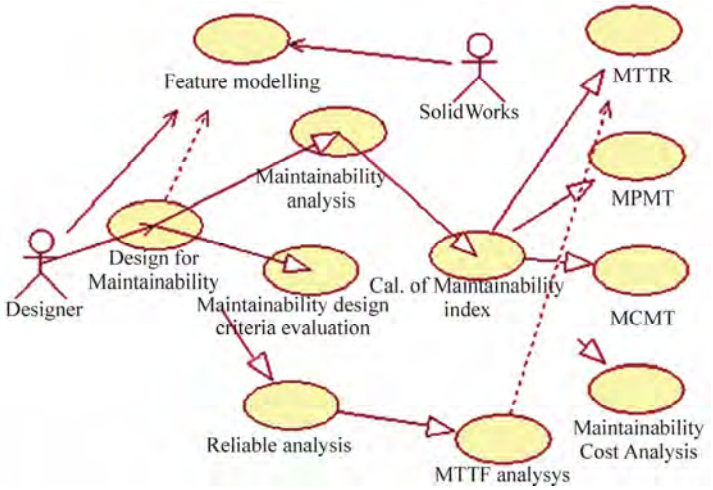


Figure 6. The use case diagram of product maintainability design support tool

4.3 The Development Process of PMDSTs

PMDSTs is a typical customization software. It can not be run without 3D CAD software SolidWorks platform. The development process of product maintainability design support software obeys following methodology.

1) Design the function of product maintainability, design support tool by using use case diagram.

2) Build main framework DLL (dynamic link library) using SolidWorks COM Add-In Wizard.

3) Build every function module DLL in C++ language.

4) Call every function module DLL in main DLL framework by using LoadLibrary method.

5) Test software function and performance by using SolidWorks Add-in interface.

The use case diagram of product maintainability design support tool is shown in Figure 6.The PMDSTs include feature modeling, design for maintainability and reliable analysis use case etc. Product feature model is built in the 3D CAD software Solidworks. Design for maintainability and reliable analysis is carried out based on the product feature modeling. The part feature information and assemble feature relation can be extracted by using Solidworks API (Application Program Interface) function.

The system is developed by using Visual C++ 6.0 and Microsoft SQL Server 2000. Product maintainability design criteria are stored in the Database. The database is connected using ODBC (Open database Connectivity). The maintainability design support software is a kind of C/S (Client/Server) software. Windows 2000 Server OS (Operation system) is installed in the server computer. Windows XP OS is installed in Client computer. The main framework DLL is created into a visual C++ DLL or visual C++ .NET DLL add-in using the SolidWorks COM Add-In Wizard included in the SolidWorks API SDK (Software Development Kit) [11].

The function StartApplication and TerminateApplication which is used to connect the solidWorks and terminate the PMDST are as follows.

```
bool CSteamTurbineSysApp::StartApplication(void)
{ // add menus to the active document
  AddMenus();
  //Add toolbars
  AddToolbars();
  // create a control item to handle application-level events
  swAppEvents* eventApp = new swAppEvents;
  eventApp->OnCreate(m_pSldWorks);
  return TRUE;
}
void CSteamTurbine-SysApp::TerminateApplication(void)
{ if (m_pSldWorks == NULL)
    return;
  // remove all menus
  RemoveMenus();
  //remove the toolbars
  RemoveToolbars();
  // release the PropertyManager object
  ReleasePage();
  LPMODELDOC pModDoc = NULL;
  HRESULT res = TheApplica-tion->GetSWApp()->get_IActiveDoc(&pModDoc);
  if( pModDoc == NULL )
   TheApplication->m_pActiveDoc = NULL;
  int count = m_EventList.GetCount();
  for (int i=0; i<count; i++)
  {       CObject* headEvent = m_EventList.GetHead();
    delete headEvent;
  if (m_pActiveDoc != NULL)
   m_pActiveDoc->Release();
```



```
    // disconnect from SolidWorks
    m_pSldWorks->Release();
    m_pSldWorks = NULL;
  }
```

Every function module DLL is called in main DLL framework by using ::LoadLibrary method is as follows.

```
  void MaintainabilityAnalysis ()
  {// MaintainabilityAnalysis interface function
    CString m_AppPath;
    m_AppPath=GetAppPath();
    CString strConnect;
    LPSLDWORKS m_pSldWorks;
  // strConnect="MTCADServer";
    m_pSldWorks= TheApplication->GetSWApp();
    LPMODELDOC2 pModDoc = NULL;
    HRESULT res =
m_pSldWorks->get_IActiveDoc2(&pModDoc);
    LPPARTDOC pPartDoc = NULL;
    res = pMod-
Doc->QueryInterface(IID_IPartDoc,(LPVOID
*)&pPartDoc);
    typedef void (WINAPI * SeriousDe-
gree)(LPMODELDOC2,LPPARTDOC,LPSLDWORKS)
;
    HINSTANCE hmod;
    hmod = ::LoadLibrary(_T("D:\\Steamturbine Main-
tenanceSoft\\ MaintainabilityAnalysis.dll"));
    if(hmod==NULL)
    {AfxMessageBox(_T("MaintainabilityAnalysis.dll
can not be found in current directory!")); }
    SeriousDegree lpproc;
    lpproc = (SeriousDegree)GetProcAddress(hmod,"
MaintainabilityAnalysis ");
    if(lpproc!=( MaintainabilityAnalysis)NULL)
     (*lpproc)(pModDoc,pPartDoc,m_pSldWorks);
    FreeLibrary(hmod);
  }
```

4.4 Maintainability Measure Index Analysis Case

The process block diagram of a steam turbine-generator system is shown in Figure 7 [12]. In the design process of steam turbine-generator system, MTTR and MTBF etc. Maintainability measure index of steam turbine-generator system is calculated as follows.

$$T_{mttr} = (\sum_{i}^{m} \lambda_i T_i) / \sum_{i}^{m} \lambda_i = \frac{39227}{370.43} = 105.9$$

$$T_{mtbf} = \frac{10^6}{\sum \lambda} = 2.699$$

$$A = \frac{T_{mtbf}}{T_{mtbf} + T_{mttr}} = 96.2$$

The calculation process of maintainability measure index of steam turbine-generator system is shown in Figure 8. Every Sub-system/assembly MTBF value must be calculated at first in order to get MTTR, MTBF and availability etc. index of steam turbine-generator system. The designer can read Solidworks 3D feature model of current subsystem by pressing 'View 3D Model' button.

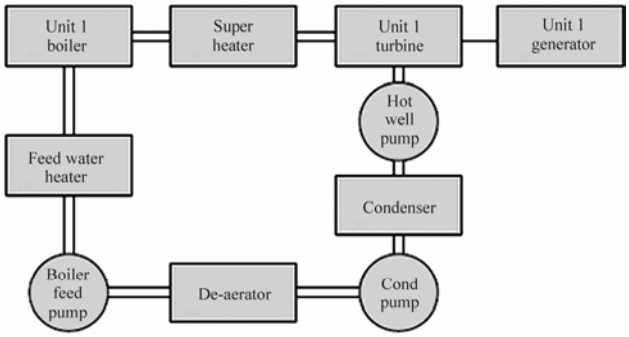


Figure 7. Process block diagram of a steam turbine-generator system

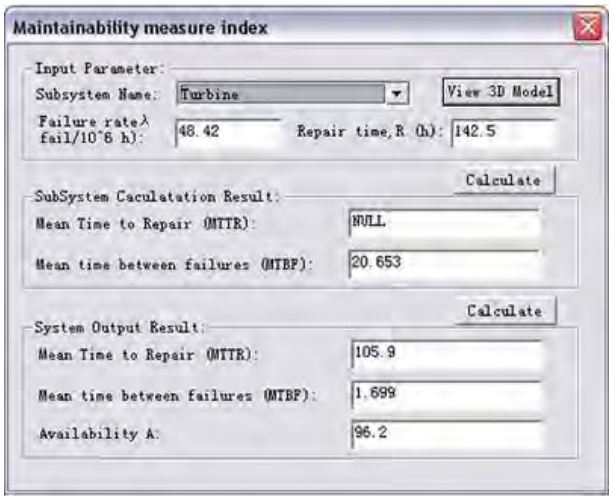


Figure 8. The calculation of maintainability measure index

5. Conclusions

In this paper, the maintainability design criteria and measure index used in product maintainability analysis are discussed. A product feature library for steam turbine design is built. The feature library can support steam turbine product feature modeling quickly. Product maintainability design method based on feature modeling can help to analyze product maintainability in the product design process. PMDSTs is developed by using Visual C++ 6.0 and Microsoft SQL Server 2000. PMDSTs can support designer to evaluate product maintainability by applying maintainability design criteria and maintainability measure index in the design stage. This method will help to enhance product maintainability efficiently. PMDSTs will be used to product computer support collaborative design (CSCD) process through the next step

of developing new collaborative support function.

6. Acknowledgment

This paper is supported by the Key Technologies R&D Program of Wuhan City (No. 200810321153）and Wuhan Youth Science and Technology Chenguang Program (No.200750731289).

REFERENCES

[1] http://www.barringer1.com/jul01prb.htm.

[2] B. Abdullah, M. S. Yusoff, and Z. M. Ripin, "Integration of design for modularity and design for assembly to enhance product maintainability," Proceddings of 1st International Conference 7th AUN/SEED-Net Fieldwise seminar on Manufacturing and Material processing, pp. 263–269, University Malaya, 2006.

[3] C. A. Slavila, C. Decreuse, and M. Ferney, "Fuzzy approach for maintainability evaluation in the design process," Concurrent Engineering, Vol. 13, No. 4, pp. 291–300, 2005.

[4] A. Coulibaly, R. Houssin and B. Mutel, "Maintainability and safety indicators at design stage for mechanical products," Computers in Industry, Vol. 59, No. 5, pp 438–449, 2008.

[5] Y. F. Ding and B.Y. Sheng, "Study on product maintenance integration model," Submit to The IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) 2009.

[6] M. Pecht, "Product reliability, maintainability, supportability handbook," CRC Press, Boca Raton, Florida, pp. 191–192, 1995.

[7] B. S. Dhillon, "Engineering maintainability," Eelservier, 2008.

[8] H. Reiche, "Life cycle cost in reliability and maintainability of electronic systems," Computer Science Press, Potomac, Maryland, pp. 3–23.1980,

[9] W. Kue, V. R. Parsad, F. A. Tillman and C. Hwang, "Optimal reliability design: Fundamentals and applications," Cambridge University Press, 2001.

[10] A. M. Sarhan, "Reliability equivalence factors of a general series–parallel system," Reliability Engineering and System Safety, Vol. 94, No. 2, pp.229–236, 2009.

[11] SolidWorks Corporation. "Solidworks 2000 API help," 2006.

[12] Stapelberg and R. Frederick, "Handbook of reliability, availability, maintainability and safety in engineering design," Springer, 2009.

J. Software Engineering & Applications, 2009, 2: 173-194
doi:10.4236/jsea.2009.23025 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Research on Software Production Support Structure

Jiangping WAN[1,2]

[1]School of Business Administration, South China University of Technology, Guangzhou, China; [2]Institute of Emerging Industrialization Development South China Univ. of Tech., Guangzhou, China.
Email: csjpwan@scut.edu.cn

Received March 11th, 2009; revised July 10th, 2009; accepted 12th, 2009.

ABSTRACT

Firstly, it is found that process design is necessary for software process improvement after analyzing its complexity. Then, research methods and concepts framework are put forward, and the research content is also provided. The findings of research, including propositions of complexity of software process, the work program of complexity of software process improvement, software enterprise model and software production support structure are clarified. Finally, the demonstration, including mindbugs (cognitive barriers) in software process and the knowledge integration support structure of quality software production, is illustrated with case study. It is concluded that the research is useful for both software production and knowledge economy in the future.

Keywords**:** *Complexity, Mindbugs, Software Process, Work Program of Complexity, Software Enterprise Model, Interactive Management, Quality Software Production, Knowledge Integration*

1. Introduction

Since the computer software is a kind of logical product, its quality improvement is difficult and complex. Many researchers are trying to reduce the hardship and the cost as well. Nowadays, it is going to focus on the software process of software production. Software process is the set of tools, methods, and practices used to produce a software product. The objectives of software process improvement (SPI) are to produce products according to the plan while simultaneously improving the organization's capability to produce better products. It is clear that a fully effective software process must consider the relationships of all the required tasks, including the tools and methods used, the skill, training, and motivation of the people involved [1–2].

An economist, Howard Baetjer, commented on the software process as following [3]: as software likes all capital, it is concrete knowledge, and because that knowledge is initially dispersed, tacit, latent, and incomplete in large measure, software development is a social learning process. The process is a dialogue in which the knowledge on the software is brought together and embodied in the software. The process enables interaction between users and designers, between users and evolving tools, and between designers and evolving tool (technology). It is an iterative process in which the evolving tool itself serves as the medium for communication, with each new round of the dialogue eliciting more useful knowledge from the people involved. It is obvious that software process is also an organizational knowledge-intensive learning process and needed to be supported with knowledge management.

Warfield argued that normal problems involved local or occasionally intermediate logics, but complex problems involved deep logic. Since deep logic is generally absent from representations, or if present was often masked by being embedded in thicket-like prose, the consequence often was under-conceptualization and under-documentation, as well as poor communication. Just as Aristotle said that logic was measure to reach knowledge, and it was necessary to enhance organizational learning with process design for knowledge management. The variety of fundamental operations that can be carried out with ideas is quite limited. Almost everything that needs to be done can be conceived as 1) generating ideas, 2) clarifying ideas, 3) structuring ideas, 4) interpreting structures of ideas, and 5) amending ideas. The limited number of “idea actions” means that the variety of processes needed can also be quite limited. One only needs to get processes for clarifying ideas, structuring ideas, and interpreting the structures produced. Design consists primarily of three types of intellectual activity: conceptualization, choice, and documentation. The implementation of design is its most concrete phase, but the failure

This research was supported by Key Project of Guangdong Province Education Office (06JDXM63002), Soft Science project of Guangdong Province(2007B070900026), NSF of China (70471091), and QualiPSo (IST- FP6-IP-034763).

of any one of these three prior types will usually assure the failure of the implementation. Intelligence, analysis, and synthesis make up conceptualization. Communication and interpretation make up documentation [4].

Interactive Management is a system designed specifically and painstakingly for the purpose of helping people resolve complexity in organizations [5]. And 20 laws of complexity were put forward [6]. A science of complexity integrates all of the material in order to show both the theory of complexity and the empirical evidence that has been accumulated to show the validity of the theory. A principal outcome of the science of complexity is called the work program of complexity (WPOC). This WPOC consists of two main steps: discovery and resolution of complexity. Discovery consists of description and diagnosis. Resolution consists of design and implementation [7]. WPOC was applied by a large cross-functional team of Ford engineers and system developers in the mid-1990s as an enabler to create an enterprise-wide information system [8].

2. Software, Software Production and Their Complexity

Recall that this title of Brooks's article is “No Silver Bullet” [9]. Brooks's theme is that the very nature of software makes it highly unlikely that a silver bullet will be discovered that will magically solve all the problems of software production, let alone help to achieve software breakthroughs that are comparable to those that have occurred with unfailing regularity in the hardware field. He divides the difficulties of software into two Aristotelian categories: **essence**, the difficulties inherent in the intrinsic nature of software, and **accidents**, the difficulties that are encountered today but are not inherent in software product. He lists four, which he terms complexity, conformity, changeability, and invisibility. In the context of his article, Brooks uses the word complex in the sense complicated or intricate. In fact, the names of all four aspects are used in their non-technical sense.

Complexity. It is an inherent property of software. Brooks points out that complexity phenomena can be described and explained in disciplines such as mathematics and physics. In contrast, if software is simplified, the whole exercise is useless; the simplifying techniques of mathematics and physics work only because the complexities of those systems are accidents, not essence, as is the case with software products.

Conformity. The first type of conformity identified by Brooks, software acquires an unnecessary degree of complexity as it has to interface with an existing system. The second type identified by Brooks, where software acquires an unnecessary degree of complexity because of the misconception that software is the component that is the most conformable. In other words, it makes the complexity of software be escalated.

Changeability. The functionality of a system embodied in its software, and functionality changes are achieved through changing the software. Boorks points out that that changeability is a property of the essence of software and an inherent problem that cannot be surmounted.

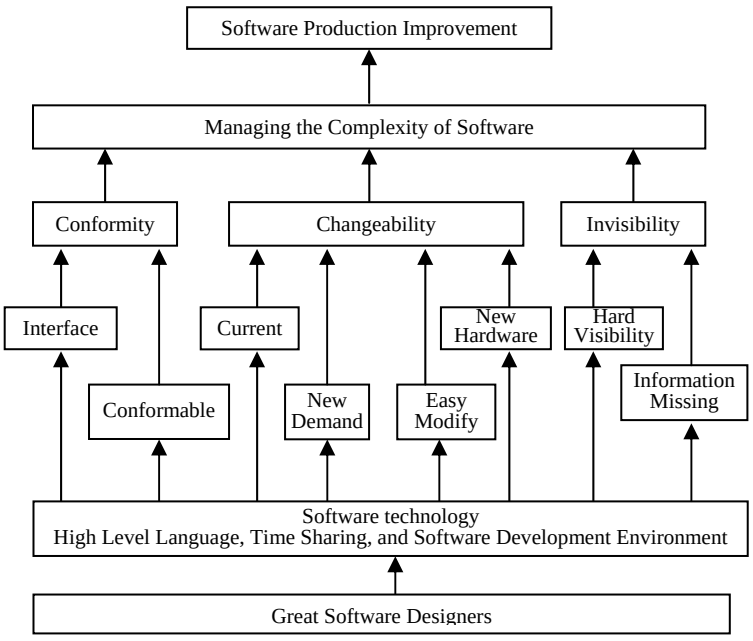


Figure 1. Approach of software production improvement by brook

Invisibility. The result of this inability to represent software visually not only makes software difficult to comprehend, it also severely hinders communication among software professionals which causes that there seems no alternative to handing a colleague a 150-page listing together with a list of modifications to be made.

Brooks considers the three major breakthroughs in software technology, namely, high level language, time sharing, and software development environment, but stresses that they solved only accidental, and not essential difficulties. For Brooks, the greatest hope of a major breakthrough in improving software production lies in training and encouraging great designers. It can be illustrated in Figure 1.

In J. N. Warfield's words [4], we could manage the complexity of software product through process design. Processes reduce personal mindbugs (cognitive barriers) primarily through the way in which information is sequenced. If members of groups are given rein to choose the topic of their discussion at random, and if several members speak at the same time, sensible discussion that leads to some organized product is hard to obtain. But if the process is designed so that the subject is broken down into a series of carefully designed questions which are presented under computer control for group discussion and resolution, the dialog becomes focused and the products of the dialog can be aggregated and organized with ease. Such a design will meet with group approval, provided the group is assured that important subjects will not thereby be excluded, and that their contributions will ultimately be incorporated on all matters perceived by group members to be relevant.

3. Research Methods

The complexity of SPI was studied through investigation on spot and literature reading. There are three problems as following: 1) What is the theory foundation of WPOC being applied to SPI? 2) How to design the WPOC of SPI? What kind of knowledge technology is necessary? 3) How to combine WPOC of SPI with the business goals of software enterprise?

The concept framework of study is established through learning Warfield's complexity theory, software engineering and quality management. The theory hypothesis is put forward that WPOC can be applied to SPI, and then the propositions of complexity of software process are educed. The propositions are theory foundation of WPOC of SPI. Basing on Warfield' WPOC, WPOC of SPI is designed. So do software enterprise model based on Microsoft's enterprise model and Infosys' knowledge management. Finally, software product support structure is established and its rationality and validity are illustrated by demonstration study [10].

The research concept framework is illustrated in Figure 2. It is established according to generating ideas, clarifying ideas, structuring ideas, interpreting structures of ideas and amending ideas. At first, the goal of software quality improvement is essential to SPI, and SPI is abstract to the theory of SPI, and the theory of SPI is structural to WPOC of SPI, and WPOC of SPI is organizational to software enterprise model and at last software enterprise model is commercial to market competition by business operation [10–18].

4. Research Content

The logical structure of the research content is illustrated in Figure 3. The contents which dashed frames represented are involved in the research. The research finds that four software essence aspects are similar to Warfield's complexity and Warfield's complexity theory can be applied to SPI to conclude five propositions of software developmental process and seven propositions of software support process whose rationality and validity are proved by successful practices of software enterprises.

Software process model regulates not only every phase task and activity in details, hierarchical sequences among tasks and activities, but also establishes the order and

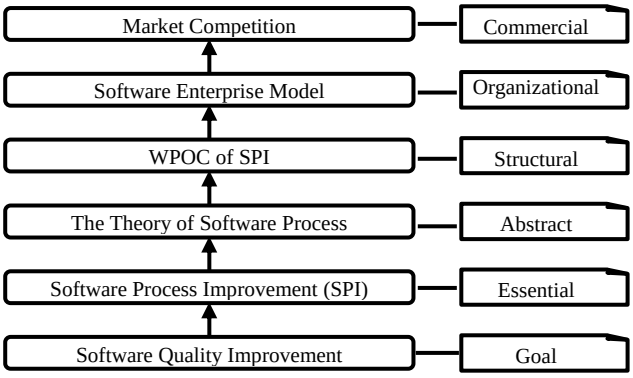


Figure 2. Diagram of concept framework

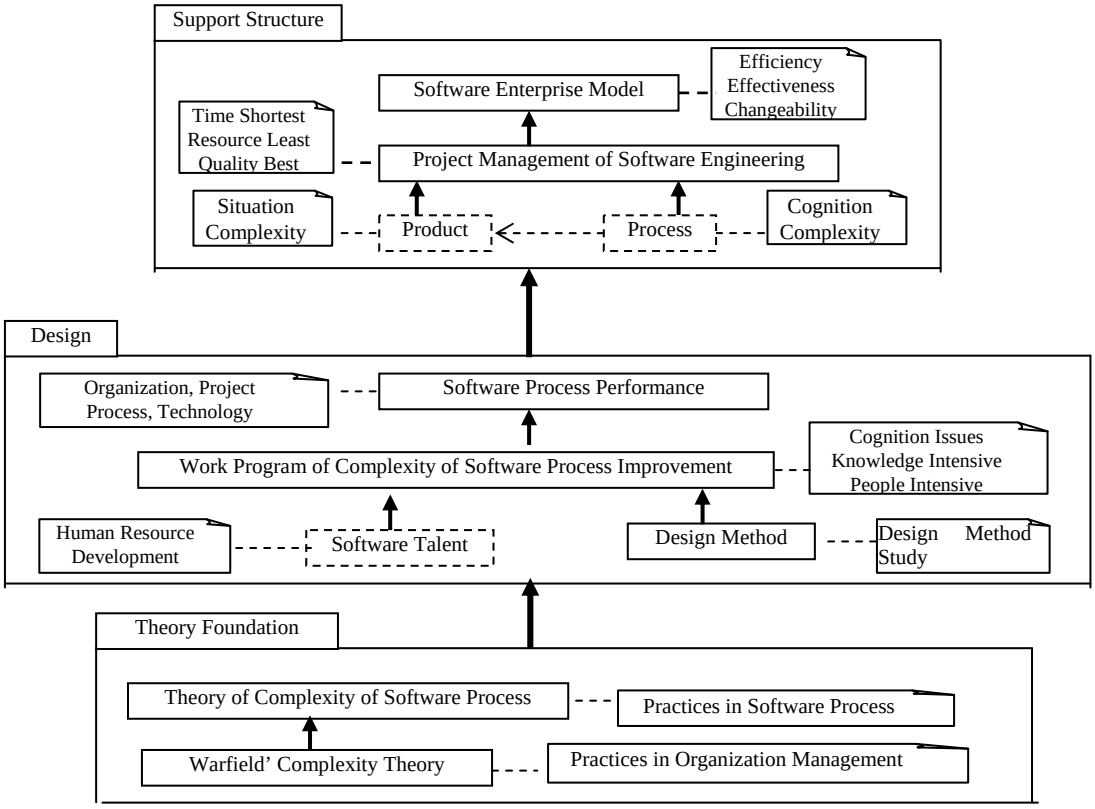


Figure 3. Diagram of the research content

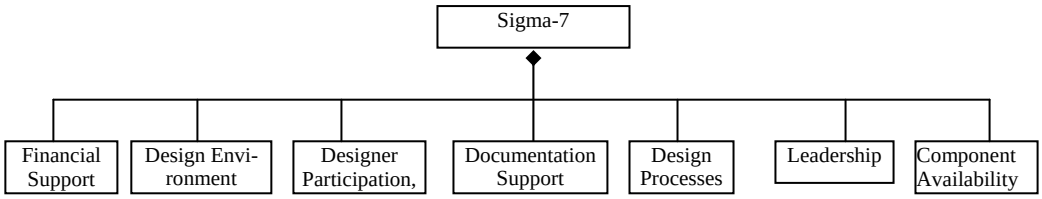


Figure 4. Class diagram of sigma-7

restrictions in every phase of software development and evolution and specifies the guideline from one phase to another. Meanwhile, it also gives the restrictions of the policy which should be followed in software development.

It is necessary to combine WPOC of SPI with software project management to realize the business goals of enterprise. The enterprise model can provide support to business operation by project because the concrete software process is determined by product including service in software engineering project management [10,18].

5. Propositions of Complexity of Software Process

The software process can be classified to software development process (SDP) and software support process (SSP) according to M. Porter's value chain model. Five complexity propositions of SDP are put forward through applying principle of generic design to seven principles of software engineering by B. W. Boehm [19]. Finally, seven complexity propositions of SSP are educed through applying principle of generic design to six principles of SPI by W. S. Humphrey [1,10].

5.1 Sigma-N Concepts

The Sigma represents the idea of integration [4], and the N represents factors to be integrated in order to achieve success. It is illustrated in Figure 4.

To make this clear, supposing that in some particular design situation, context makes it clear that financial support and component availability are readily available, but the critical factors of leadership, design environment, designer participation, documentation support and design processes have not been integrated. Then we would say

that we are dealing with a potential Sigma-5 situation. It has been found in practice that the Sigma-7 and Sigma-5 concepts are the most significant, with other numerical values such as Sigma-1 and Sigma-2 representing situations that are normally ineffective in dealing with complex issues.

5.2 Five Complexity Propositions of Software Development Process

The basic principles of software engineering by B. W. Boehm are in the following [19]:

Principle of plan (BP1): Manage using a phased life-cycle plan.

Principle of review (BP2): Perform continuous validation.

Principle of control (BP3): Maintain disciplined product control.

Principle of technology (BP4): Use modern programming practices.

Principle of visualization (BP5): Maintain clear accountability for result.

Principle of performance (BP6): Use better and fewer people.

Principle of process (BP7): Maintain a commitment to improve the process.

The relationship of six principles of software engineering with Sigma-5 are illustrated in Figure 5.

The principles that have related with design environment include BP4, BP6 and BP7.

The principles that have related with designer participation include BP4, BP6 and BP7.

The principles that have related with documentation support include BP1, BP2, BP3, BP4, BP5, BP6 and BP7.

The principles that have related with design processes include BP1, BP2, BP3, BP4, BP5, BP6 and BP7.

The principles that have related with leadership include BP1, BP3, BP4, BP6 and BP7.

There are two ubiquitously critical factors among Sigma-5: documentation support and design processes. It is necessary for software engineering to design processes of reducing the cognitive barriers (mindbugs) to improve intelligence activities' efficiency and quality, and visualize intellectual activities for management.

There are three ubiquitously principles among the seven principles of software engineering: technology principle, process principle, and performance principle. It is necessary for software engineering to learn new technology and make continuous improvement because of intelligence-intensive teamwork.

Five propositions of complexity of SDP are as follow (Figure 6): 1) Proposition of capability (SDP1). SDP is required continuously organizational learning in order to improve SDP capability of organization; 2) proposition of quality (SDP2). SDP is required to have high quality products and services through efficient cooperation; 3) Proposition of process (SDP3). SDP is required to adopt carefully design processes in order to reduce personal cognitive burden and improve group cognitive activities effectively and efficiently; 4) Proposition of documentation (SDP4).

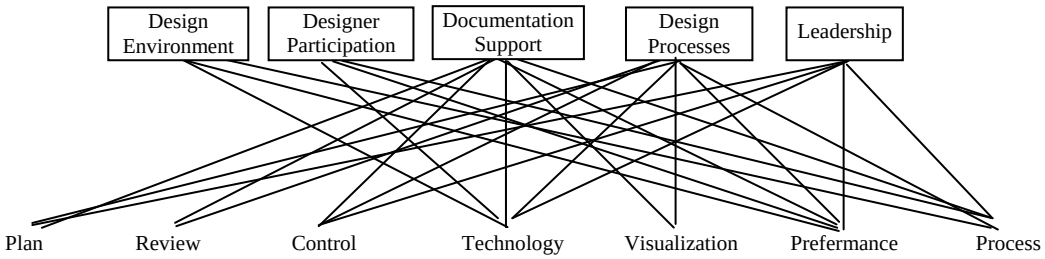


Figure 5. Relationship seven principles with sigma-5

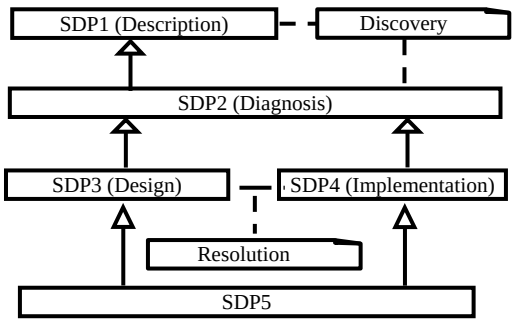


Figure 6. Five propositions of complexity of SDP

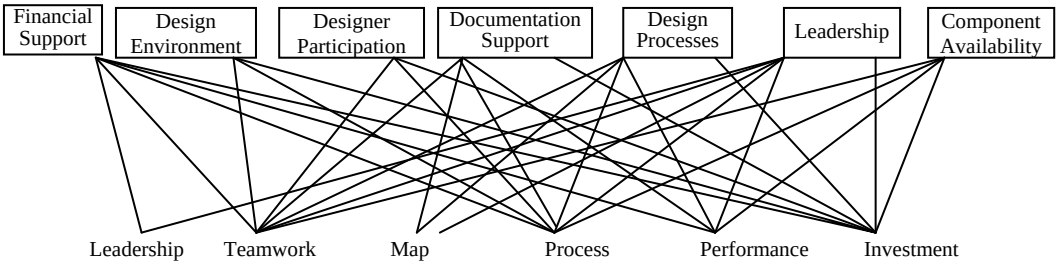


Figure 7. Relationship six principles of SPI with sigma-7

SDP is required to make documentation to communicate and interpret to visualize activities of SDP in order to control them; 5) Proposition of technology (SDP5). SDP is required to use modern software technology. The SDP1 (capability) and SDP2 (quality) are mostly based on behavioral science and goals of software engineering, but SDP3 (process), SDP4 (documentation) and SDP5 (technology) are mostly based on deep logic and software formal technology, their relationship are illustrated in Figure 6 (The hollow triangle represents inherited relation, e.g. if the child is similar to his parent, the child inherits his parent's characters. Here is to embody the proposition. e. g. the SDP1 is embodied in SDP2, SDP2 is embodied in both SDP3 and SDP4, and both SDP3 and SDP4 are embodied in SDP5. The hierarchy of concept is analyzed with object technology here, the proposition is a kind of abstract concept, and the embodying can be considered as implementation of lower abstraction). It is necessary to apply WPOC to software engineering with the propositions of SDP. Discovery is needed by SDP1 (description) and SDP2 (diagnosis), and resolution is needed by SDP3 (design) and SDP4 (implementation). WPOC must apply modern software technology according to SDP5.

5.3 Seven Complexity Propositions of Software Support Process

The basic principles of SPI by Watts S. Humphrey are in the following [1]:

Principle of leadership (HP1). Major changes to the software process must start at the top. Senior management leadership is required to launch the change effort and to provide continuing resources and priority.

Principle of teamwork (HP2). Ultimately, everyone must be involved. Software engineering is a team effort, and anyone who does not participate in improvement will miss the benefits and may even inhibit progress.

Principle of map (HP3). Effective change requires a goal and knowledge of the current process. To use a map, you must know where you are.

Principle of process (HP4). Change is continuous. Software process improvement is not a one-shot effort; it involves continual learning and growth.

Principle of performance (HP5). Software process changes will not be retained without conscious effort and periodic reinforcement.

Principle of investment (HP6). Software process improvement requires investment. It takes planning, dedicated people, management time, and capital investment.

The relationship six principles of SPI with Sigma-7 are illustrated in Figure 7.

The principles that have related with financial support include HP1, HP2, HP4, HP5 and HP6.

The principles that have related with design environment include HP2, HP4 and HP6.

The principles that have related with designer participation include HP2, HP4 and HP6.

The principles that have related with documentation support include HP2, HP3, HP4, HP5 and HP6.

The principles that have related with design processes include HP2, HP3, HP4, HP5 and HP6.

The principles that have related with leadership include HP1, HP2, HP3, HP4, HP5 and HP6.

The principles that have related with component availability include HP2, HP4, HP5 and HP6.

There are four ubiquitously critical factors among Sigma-7: financial support, documentation support, design processes and leadership. It is necessary for SPI to design processes through documentation with financial support.

There are four ubiquitously principles among the six principles of SPI: team principle, process principle, performance principle and investment principle. It is necessary for SPI to organize teamwork with carefully design processes through both excellent leadership and enough investment to get satisfactory performance.

Seven propositions of complexity of SSP (Figure 8) are in the following: 1) Proposition of capability (SSP1). SSP is required to make continuously organizational learning in order to improve software process capability of organization; 2) Proposition of performance (SSP2). SSP will not be retained without conscious effort and periodic reinforcement; 3) Proposition of leadership (SSP3). SPI must start from the top. Senior management leadership is required to launch and participate in SSP; 4) Proposition of process (SSP4). SSP is required to adopt carefully design processes in order to reduce personal cognitive burden

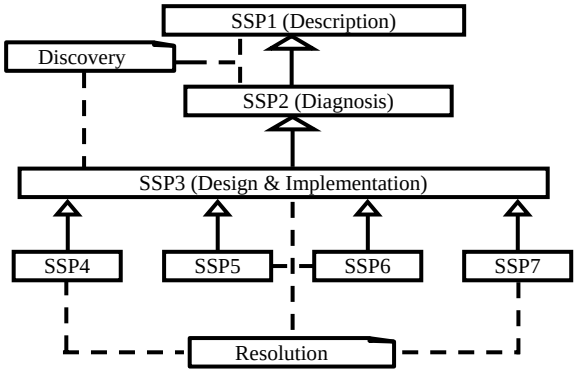


Figure 8. Seven propositions of complexity of SSP

and improve group cognitive activities effectively and efficiently; 5) Proposition of documentation (SSP5). SSP is required to make documentation to communicate and interpret for visualizing activities of SPI to control them; 6) Proposition of teamwork (SSP6). Everyone must be involved. SSP is a team effort, and anyone who does not participate in SPI will miss the benefits and may even inhibit progress; 7) Proposition of investment (SSP7). SSP is required to be projected, assigned full time stuffs, managed time and to be invested.

Watts S. Humphrey said that the key topics to focus on once the decision has been made to invest in process improvement are in the following [1]: 1) To improve the software process, someone must work on it (SSP3, SSP6, SSP7); 2) Unplanned process improvement is wishful thinking (SSP4, SSP5); 3) Automatic of a poorly defined process will process poorly defined results(SSP1, SSP2, SSP3); 4) Improvements should be made in small, tested steps(SSP1, SSP2, SSP3); 5) Train, train, train(SSP1, SSP3, SSP7).

According to a systematic survey of CMM experience and results by SEI (Software Engineering of Institute), there are seven empirical principles of successful SPI in the following [2]:

EP1. Management control is tightly related with the successful SPI (SSP2, SSP4, SSP5).

EP2. To establish software process group, which includes technicians, correctly, and assign SPI responsibility clearly and enough (SSP4, SSP6).

EP3. To take SPI carefully and retain experienced specialists and refer outside seriously (SSP1, SSP7).

EP4. To estimate the progress and cost of engineering correctly (SSP1, SSP2, SSP4, SSP7).

EP5. Managers consider seriously the influences, which come from software organizational culture, external business environment and internal environment, on SPI (SSP1, SSP2, SSP3).

EP6. To conduct operation of software correctly (SSP4, SSP5, SSP6, SSP7).

EP7. To establish software quality plan perfectly (SSP1, SSP2, SSP3, SSP4, SSP5, SSP6, SSP7).

6. Work Program of the Complexity of Software Process Improvement

The WPOC of SPI consists of two phases (Figure 9): discovery and resolution [10].

Discovery includes two steps: 1) Describing the software process, the basic activities of software process are described and their mindbugs are identified and classified. It is necessary to consider the mindbugs, because software process is usually a complexity group cognitive process. Object technology, a kind of knowledge technology, is useful since it conforms to human cognitive habits and can reduce cognitive burden effectively. It suggests that the basic activities of software process are firstly described with object technology to understand enough the architecture and concretely activities in details, the mindbugs of software process are identified and classified for diagnosing software process. 2) Diagnosing the software process, Software Capability Maturity Model (SW-CMM) by SEI is an effective method and tool. SW-CMM can be analyzed with object technology to understand its essence that software process performance is the core of SW-CMM and SW-CMM can be used to illuminate performance structure and target of software process.

Resolution also consists of two steps. 1) Designing SPI, three popular SPI models are IDEAL model (process model includes Initial, Diagnosis, Establishment, Action and Learning five steps and is a de fact implementation methodology of SW-CMM) by the HP corporation and SEI, three-phrase method (product model includes Understanding, Assessing, and Packaging three steps) by

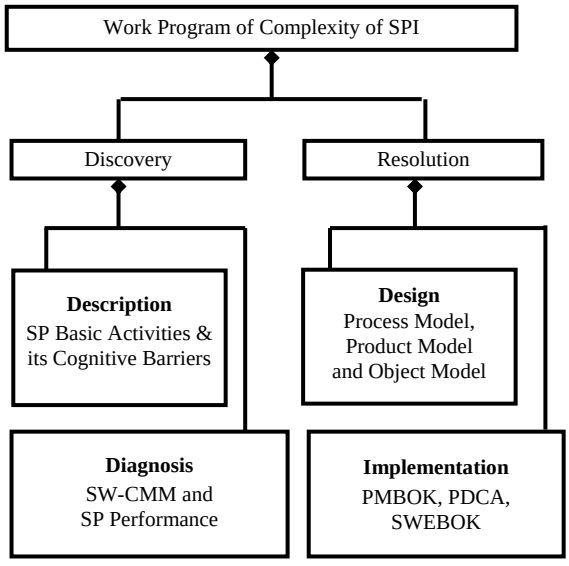


Figure 9. Work program of complexity of SPI

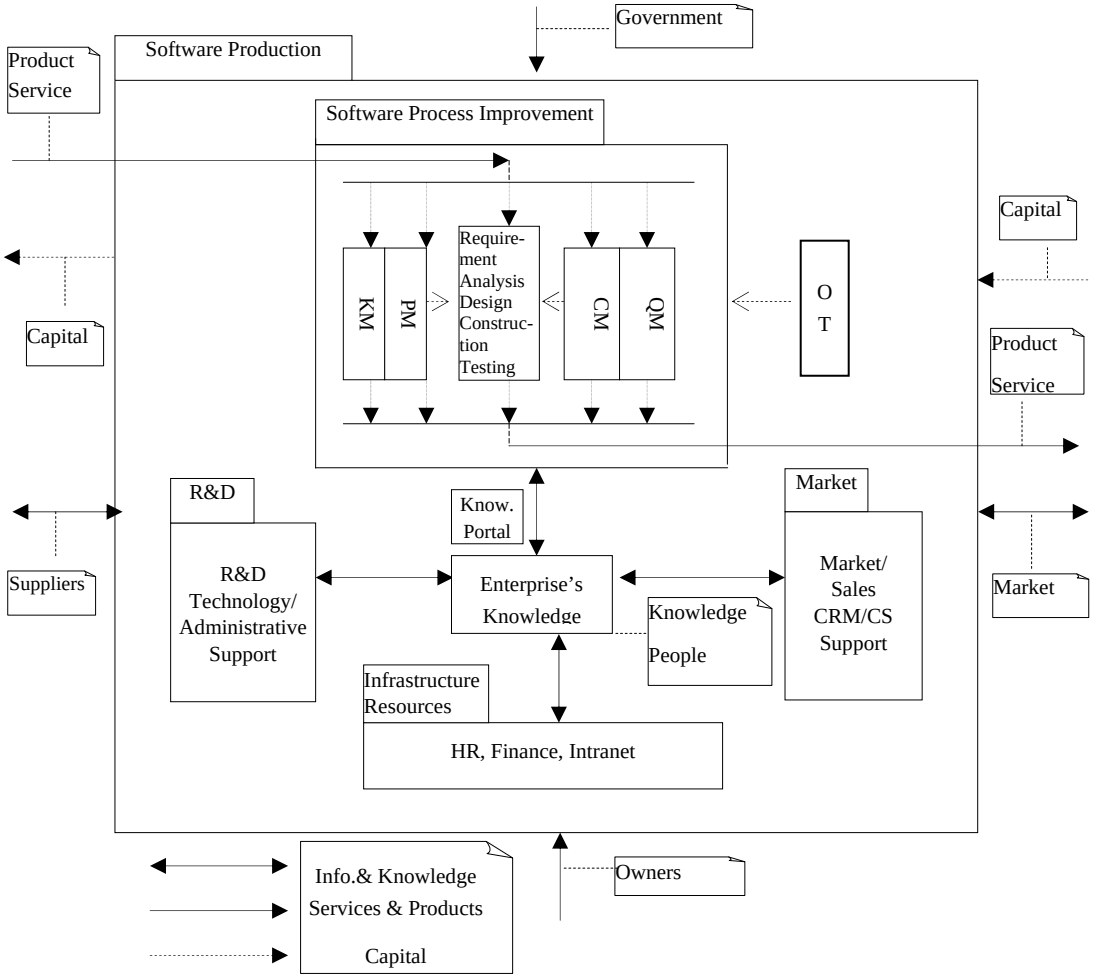


Figure 10. Knowledge enabling enterprise model for software production

NASA, and Rational Unified Process (object model includes Inception, Elaboration, Construction and Transition four phases and nine core process workflows, which are business modeling, requirement, analysis and design, implementation, testing, project management, configuration, change management and environment etc.). It is important to understand the essence that there are both merits and demerits in three models. When designing software processes, it is necessary to actively think about real contexts and issues in order to apply the models in a creative way, not in a rote way. 2) Implementation SPI. The first important knowledge is project management, which is often achieved with half effort to do SPI according to project management in real world. The second is method of quality management used in traditional industry, e.g. PCDA cycle by W. E. Deming, which is used in SPI by NEC do Brazil (NDB), but the traditional industry is quite different from software industry, which is knowledge-intensive and brain-intensive, so IEEE SWEBOK (Software Engineering Body of Knowledge) is necessary. These are knowledge technologies needed to implement SPI.

7. Knowledge Enabling Enterprise Model for Software Production

A model is an abstract representation of reality expressed in terms of some formalism. An enterprise model may be used for various purposes such as facilities design, systems architecture, organization, simulation, optimization, performance measurement and benchmarking. Here it will be used to describe and understand the operational of a software organization [13,18].

A knowledge enabling model for software enterprise with generic structure is illustrated in Figure 10. The notation of UML is used. The outer boundary frames business operation. Inputs are services and products, capital, information and knowledge. Outputs are services and products, capital, information and knowledge. Governmental regulations and other frame conditions may be regarded as control. Owners and external organizations

comprise the mechanism. Software production box includes market, sale, Customer Relationship Management (CRM) and Customer Service (CS). Research and Development (R&D), technological and administrative support functions, infrastructure resources (any kind, e.g. Human Resources and Intranet etc.), organizational knowledge portal (the knowledge keys to successfully software project), as well as SPI. The market is goal, while sale is means and CRM & CS are assurance. R&D is the core element, technological and administrative supporting are interfaces and infrastructure resources. The inner box, software process improvement (development), includes software process (inception, elaboration, construction and transition, them are not illustrated in the Figure 10) and core workflows and activities supporting. The core workflows are in the following: requirement, analysis, design, construction and testing. All of the development activities are architecture based (OT, object technology), including configuration (change) management (CM), project management (PM, plan, control, cooperation and standard etc.), quality management (QM, including analysis and assurance), knowledge management (KM) in SPI, Knowledge portal is facility for knowledge accumulating, sharing and communication in organization scope, KM should be supported by HR, Finance, Intranet etc.

8. Software Production Support Structure

Basing on interactive management support structure [5,12], software production support structure can be illustrated in Figure 11.

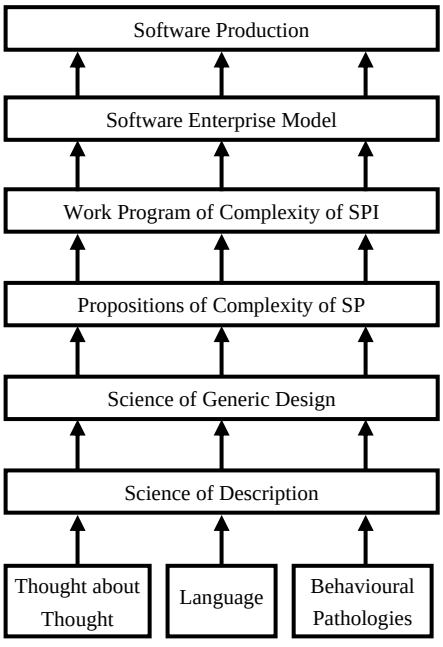


Figure 11. Software production support structure

It consists of boxes that contain text and arrows that connect the boxes. The arrows are all oriented upward. They show the directional flow of support**.** If there is a directed path from some box to some higher level box, the one below supports the one at the other end of the path. This is the basic rule for understanding this support structure. The word "support" is the key point. It means that you will find necessary information in the lower box for the development of what appears above. In this instance, you can see, for example, that software production support structure is supported by two sciences and one theory and one process and one model. The two sciences are all supported by these three principal components: 1) **Thought about thought** (which J. N. Warfield called "second-order thought"); 2) **Language**; 3) **Behavioural pathologies** (which is the information about human limitations, individually, in small groups, and in larger organizations; the information coming from the research in psychology, sociology, and organizational studies).

The date 350 B.C. corresponded roughly to the period of Aristotle's work in logic, in which he developed the first formal statement of deductive logic, along with the concept of categorizing topics so that it would be possible to work with and compare collections of ideas, as well as with the individual ideas themselves. Prof. Warfield has described the pattern that evolved over the many centuries that brought thought about thought into our view today, so that those ideas find their ways into SPI, where they help people resolve complexity of SPI. The idea is applied to Chinese enterprise on the condition that Warfield's theory must be combined with Chinese traditional culture and thought, such as Change of Book and "Shili-Wuli-Renli" etc.

A Science of Generic Design [4]: This science focused upon the use of information coming from a high- quality description to develop a set of options from which an alternative design could be chosen for implementation. It focuses upon the design of socio-technical systems. Software engineering, which is also socio- technical systems, can be applied the science of generic design. The propositions of SDP and SSP are educed and become the theory foundation of SPI. This theory can guide the design of WPOC of SPI (structural presentation).

When working on a software project, software engineers apply principles based on the foundation to software developing processes and the product through forming specific and pragmatic methods and tools [18]. It is necessary for complexity interactive cognition processes because that SPI should be combined with software project management, which is needed to conform to enterprise business goals. Software enterprise model can be considered as an application model of e-business for software enterprise [10, 15, 18, 20].

9. Demonstration

Demonstration includes two parts as following: 1) Prof. J. N. Warfield has tried to identify and name each distinctive origin of one or more behaviorally related symptoms. Mindbugs (cognitive barriers) according to SW- CMM are studied. The research finds that the most common mindbugs of SPI is misattribution of consensus. 2) The knowledge integration support structure of quality software production is illustrated with case study.

9.1 Mindbugs in Software Process

9.1.1 The Identification and Classification of Mindbugs

Warfield have tried to identify and name each distinctive origin of one or more behaviorally-related symptoms (as "mindbugs" to bring the language in line with contemporary computer languages) [21]. So far, twenty-five mindbugs have been identified, which are grouped into four categories: Mindbugs of Minsinterpretation: those where concepts are misconstrued or misattributed, because of faulty interpretation, Type M; Mindbugs of Clanthink: those where concepts are very widely perceived to be correct, but are demonstrably incorrect, Type C; Mindbugs of Habit: those which involve ingrained behavior, evinced with essentially no conscious thought, Type H; and Mindbugs of Error: just plain mistakes, Type E. It can be summarised in Table1 (Where a Mindbugs is at least assigned to more than one type, and several types are separately acknowledged).

Table 1. The identification and classification of mindbugs

M	C	H	E
M1, Misinterpretation of Linguistic Adequacy of Natural Language , C1	C1, Misinterpretation of Linguistic Adequacy of Natural Language, M1	H1, Indistinguished Affinity to Unstructured Discussion, C6, E3	E1, Susceptibility to the Fad of the Month, H6
M2**,** Misinterpretation of Linguistic Adequacy of Object Languages, C2	C2, Misinterpretation of Linguistic Adequacy of Object Languages, M2	H2, Adversity to Budgeting for Interface Expenses, C7, E4	E2, Unawareness of Imputed Structure, H10
M3, Confusing Prestige with Authoritativeness	C3, Misconstruing Technology as Science (and vice versa), M4	H3, Affinity to All-Encompassing Dichotomies	E3, Indistinguished Affinity to Unstructured Discussion, C6,H1
M4, Misconstruing Technology as Science (and vice versa), C3	C4, Insensitivity to Conceptual Scale	H4, Leaping to Misassociation	E4, Adversity to Budgeting for Interface Expenses, C7, H2
M5, Misconstruing Structural Incompetence as Innate Incompetence	C5, Insensitivity to the Presence and Origins of Human Fallibility	H5, Insensitivity to Role Distinction	E5, Mistaken Sense of Uniqueness
M6, Misattribution of Consensus	C6, Indistinguished Affinity to Unstructured Discussion, H1,E3	H6, Susceptibility to the Fad of the Month, E1	E6, Mistaken Sense of Similarity
M7, Misconstruing Persistence as Validity	C7, Adversity to Budgeting for Interface Expenses, H2, E4	H7, Insensitivity to the Significance of Information Flow Rates	E7, Misconstruing Philosophy as Ideology(and vice versa)
		H8, Adversity to Deep Though	E8, Misassignment of Relative Saliency
		H9 Failure to Distinguish among Context, Context, and Process	E9, Irresponsible Propagation of Underconceptualized Themes
		H10, Unawareness of Imputed Structure, E2	E10, Unawareness of the Cumulative Impact of Many Collocated Mindbugs

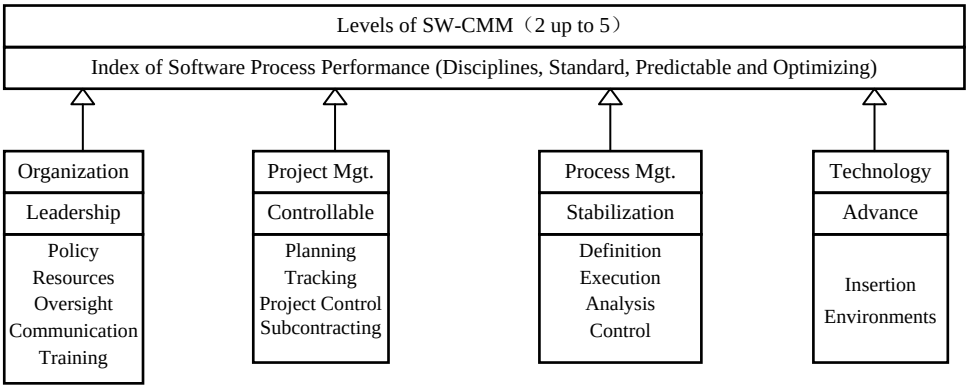


Figure 12. Class diagram of index system of software process performance

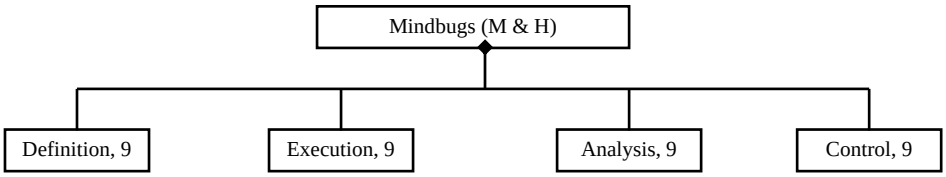


Figure 13. Class diagram of mindbugs of process management of SPI

Table 2. The reliability of mindbugs questionnaire

Mindbugs	P1	P2	P3	P4	P5	P6	P7	P8	P9
Reliability	0.8447	0.8816	0.8570	0.8701	0.8882	0.9190	0.9031	0.9185	0.8934

9.1.2 Framework of Software Capability Maturity Model

SPI actions are divided into 15 categories, which are further grouped into 4 major topics, namely organization, project management, process management and technology. They are summarised in Figure 12 (The four views are integrated into one figure to illustrate that the four views is a whole) [1].

9.1.3 Research Design

The five levels Likert scale is taken. The process management of SPI is selected referring to key areas of SW-CMM. The process management of SPI involves process definition, process execution, data gathering and analysis, and process control. The process definition provides a standardized framework for task implementation, evaluation, and improvement. Process execution defines the methods and techniques used to produce quality products. Analysis deals with the measurements, including software products and processes, and uses this data. Process control concerns the establishment of mechanisms to assure the performance of the defined process and process monitoring and adjustment where improvements are needed [1].

The major mindbugs of SPI are mindbugs of M and mindbugs of H through literature review and investigation on spot. The overlap mindbugs of M and H are removed. There are nine mindbugs in the following to be selected for study: Confusing Prestige with Authoritativeness (M3, P1), Misconstruing Structural Incompetence as Innate Incompetence (M5, P2), Misattribution of Consensus (M6, P3), Misconstruing Persistence as Validity (M7, P4), Leaping to Misassociation (H4, P5), Insensitivity to Role Distinction (H5, P6), Insensitivity to the Significance of Information Flow Rates (H7, P7), Adversity to Deep Though (H8, P8), and Failure to Distinguish among Context, Context, and Process (H9, P9). The first four mindbugs are in the type of minsinterpretation, and others of habit. The Pi (I=1, 9) is taken for easy documentation, It can be illustrated in Figure 13.

9.1.4 Statistic Analysis

The questionnaire survey was carried in Guangdong Province of China, including Center of BSS&OSS Support of China Telecom Corporation Ltd. Guangzhou Research Institute, Zhongshan Mobile Telecom Corporation, China Industry and Commercial Bank Guangzhou Branch, Asia Info et al. The number of the total questionnaire is 59, and there are 41 effective ones.

According to the above nine kinds of mindbugs, reliability analysis result is illustrated in Table 2. P6 is the top one (0.9190) and P1 is at the bottom (0.8447), but all

are bigger than 0.8. The internal consistency of questionnaire is very good.

Table 3 illustrates the following information:

a) The minimum of mindbugs are all 1, except P8 (1.25). The maximum of first five mindbugs are all 5. The other mindbugs are in the following respectively: the P6 is 4.00, and the P7 is 4.75, and the P8 is 4.00, and the P9 is 4.75.

b) The maximum mean is P3 (3.4634), and the minimum mean is P6 (2.7966), the average of the means is about 3. The descending order is as follows: P3 (3.4634), P4(3.2561), P2(3.2012) and P1(3.1890). Every type M of mindbugs is above the type H of mindugs.

One-sample T Test is taken for every hypothesis (There is some kind of mindbugs in process management of SPI). Test Value sets 3 (basing on descriptive statistics) and 95% confidence interval of the difference is taken, The T test is passed. The result is illustrated in Table 4.

a) The mean of the mindbugs of M is 3.2774, and the mean of the mindbugs of H is 2.9707.

b) In the process definition, the mean of mindbugs of M is 3.2256 and the mean of the mindbugs of H is 2.9707. In the process execution, the mean of the mindbugs of M is 3.2500 and the mean of the mindbugs of H is 2.9951. In the process analysis, the mean of the mindbugs of M is 3.3049 and the mean of the mindbugs of H is 2.9707. In the process control, the mean of the mindbugs of M is 3.3293 and the mean of the mindbugs of H is 2.9463. The biggest of mindbugs of M is the process control, and the following is process execution. The biggest of mindbugs of H is the process execution, and the following are both process definition and process analysis. The mindbugs of M is more serious than the Mindbugs of H. These are consistent with the results of both literature review and field study

It is necessary to research on SPI in the view of cognition. The research finds that the most common mindbugs of SPI is Misattribution of Consensus (M6), and the finding is consistent with literature review and investigation on spot. It is suggested to resolve the mindbugs in Type M, including M6 in the first place, then framebreak and remodel to resolve the mindbugs of Type H in SPI.

Table 5 illustrates the following information:

Table 3. The descriptive statistics of mindbugs

Mindbugs	N	Minimum	Maximum	Mean	Std. Deviation
Confusing Prestige with Authoritativeness (P1)	41	1.00	5.00	3.1890	.9300
Misconstruing Structural Incompetence as Innate Incompetence (P2)	41	1.00	5.00	3.2012	.9000
Misattribution of Consensus (P3)	41	1.00	5.00	3.4634	.8112
Misconstruing Persistence as Validity (P4)	41	1.00	5.00	3.2561	.7593
Leaping to Misassociation (P5)	41	1.00	5.00	3.0122	.9202
Insensitivity to Role Distinction (P6)	41	1.00	4.00	2.7866	.9962
Insensitivity to the Significance of Information Flow Rates (P7)	41	1.00	4.75	3.0427	.9697
Aversity to Deep Though (P8)	41	1.25	4.00	3.0000	.9066
Failure to Distinguish among Context, Context, and Process (P9)	41	1.00	4.25	3.0122	.8695

Table 4. One-sample T test

	Test Value = 3				95% Confidence	
	t	df	Sig. (2-tailed)	Mean Difference	Interval of the Difference	
Mindbugs					Lower	Upper
Confusing Prestige with Authoritativeness(P1)	1.301	40	.201	.1890	-.1045	.4826
Misconstruing Structural Incompetence as Innate Incompetence (P2)	1.432	40	.160	.2012	-8.2866E-02	.4853
Misattribution of Consensus (P3)	3.658	40	.001	.4634	.2074	.7195
Misconstruing Persistence as Validity (P4)	2.160	40	.037	.2561	1.644E-02	.4958
Leaping to Misassociation (P5)	.085	40	.933	1.220E-02	-.2782	.3026
Insensitivity to Role Distinction (P6)	-1.372	40	.178	-.2134	-.5278	.1010
Insensitivity to the Significance of Information Flow Rates (P7)	.282	40	.780	4.268E-02	-.2634	.3488
Aversity to Deep Though (P8)	.000	40	1.000	.0000	-.2861	.2861
Failure to Distinguish among Context, Context, and Process (P9)	.090	40	.929	1.220E-02	-.2623	.2867

Table 5. Compare minsinterpretation mindbugs with habit mindbugs

	N	Minimum	Maximum	Mean	Std. Deviation
Process Definition (M)	41	1.00	5.00	3.2256	.8020
Process Definition (H)	41	1.00	4.40	2.9707	.8241
Process Execution (M)	41	1.00	5.00	3.2500	.8422
Process Execution (H)	41	1.00	4.20	2.9951	.8781
Process Analysis (M)	41	1.00	5.00	3.3049	.7876
Process Analysis (H)	41	1.20	4.20	2.9707	.8301
Process Control (M)	41	1.00	5.00	3.3293	.8449
Process Control (H)	41	1.20	4.00	2.9463	.8025
Means (M)	41	1.00	5.00	3.2774	.7514
Means (H)	41	1.20	4.10	2.9707	.7943

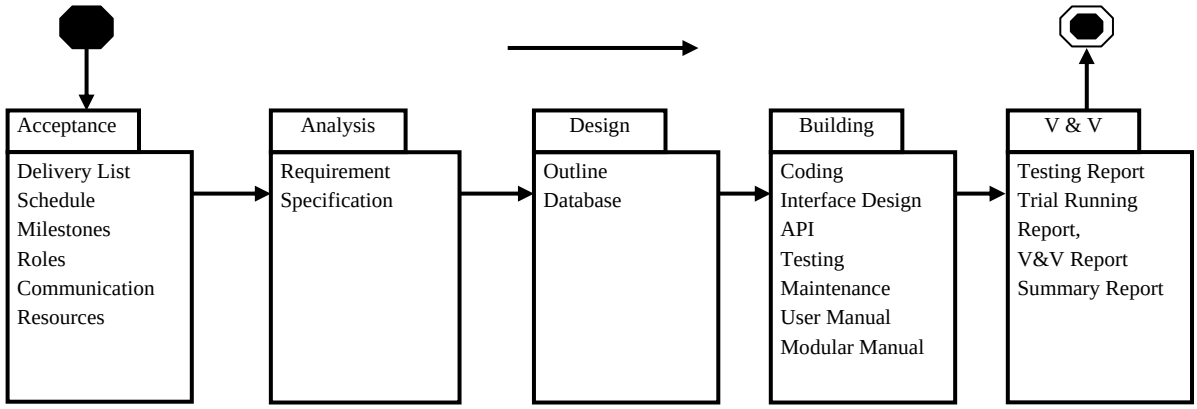


Figure 14. Software project process model of XXX corp

9.2 Case Study

XXX Corp. was established in 1995 and is a high-tech enterprise of Guangzhou. Its mindbugs in software processes, software project organization, and keys to software project management are analyzed. Finally, knowledge integration support structure of quality software production is put forward.

9.2.1 Primary Activities of Software Process

Software project usually is carried through five steps in the following: (1) Acceptance commission; (2) Analyzing requirement; (3) Design outline; (4) Building; (5) Verification and validation, It can be illustrated in Figure 14.

(1) Acceptance commission. Including pre-sales, investigation, requirement investigation, submitting requirement investigation report, feasibility study report, assignment software development tasks; (2) Analyzing requirement. Requirement specification, which is V&V to be effective, is submitted. It is both the baseline of software engineering management and the most importance gist for check and acceptance. (3) Design outline. Including outline design and database design. (4) Building. Detailed design based outline design according to coding specification, interface design specification and API specification etc. (5) Verification and Validation. Check and accept the system, including a series of testing, trial running and evaluation, submitting testing analysis report, trial running report, check and accept report and project development summary report.

9.2.2 Mindbugs in Software Process

Because of the programmers failing to rationally organizing the development team, there are problems in the following: (1) Working following feeling in the first place, working following the operator's scattered ideas in the end (no well-done requirement description, Type M); (2) Think hard, working while thinking, including writing documentation (no well-done project plan, Type C & H); (3) Think hard without guidance, it is often to discover that the system couldn't be integrated at last (no well-done process specification, Type H & E); (4) It often lose money in business when the project is finished (no well-done milestones, four types). The programmers often feel weariness, distress, at sea and no achievement (complexity, mindbugs). The phenomena are very com-

mon among software enterprises in China.

9.2.3 Software Project Organization

Developing organization scientifically is the foundation of highly efficient and stable project operating. In accordance with the CMM model, the company established SEPG (Software Engineering Process Group, similar to the Technical Group), SQA (Software Quality Assurance) and SEG (Software Engineering Group), and this led to form a balanced system including the legislation, supervision and law enforcement and realize the organization theme of the software process improvement performance model. It is illustrated in Figure 15. By 2002, the company had developed relatively complete and formal documents to manage software processes, and set up a knowledge management center.

9.2.3.1 Technical Group

In the software development processes, SEG has to obtain large amount of technical resources which relies on the day-to-day accumulation and constantly bringing in external technical resources. At the same time, by establishing development standards and norms, and completing technical proposal verification and technical personnel evaluation as well as the evaluation of researches which do not belong to any R&D development team, the development team's working efficiency can be assured and duplication of enterprises investment can be avoided. Consequently, the technical resources can be fully used. It can be illustrated in Figure 16.

The characteristics of the technical group include: 1) involving the functions of SEPG; 2) never directly undertaking the entire project tasks; 3) mastering the core technology, managing the technical capability of enterprises, and achieving processes management and technology introduction.

The functions of the technical group include: 1) technical standards: drawing up technical standards and specifications to form a unified internal technical language which is conducive to initial learning, technology communication and mutual supports among the teams. 2) technology accumulation: drafting technical plans, ensuring there are plans to introduce new technologies and organizational research and development so as to realize technology sharing as well as assessment and enhancement of the technological capability.3) knowledge resuing: managing the achievement of technology and knowledge base to ensure maximum reuse of knowledge and shortening the development cycle. 4) the quality of personnel: organizing technology training, technology communication and technology assessment to speed up the improvement of the quality of personnel. 5) technology quality: organizing technical evaluation and proposal verification to ensure the quality of technology.

The positions of the technical group include: 1) Technical manager is the person who is in charge of technology planning, technology assessment and the development of technology management system. 2) System analyst is responsible for technical analysis, program review and project assessment. 3) R & D engineer is responsible for the development of public components. 4) File Manager manages product base and shares modules base, knowledge base and database.

Many companies are not sizeable enough to establish the group, but these duties are so important that they must be managed by somebody who can be substituted by virtual teams composed of the department manager, file managers, some technical experts. However, the results are often not so satisfactory.

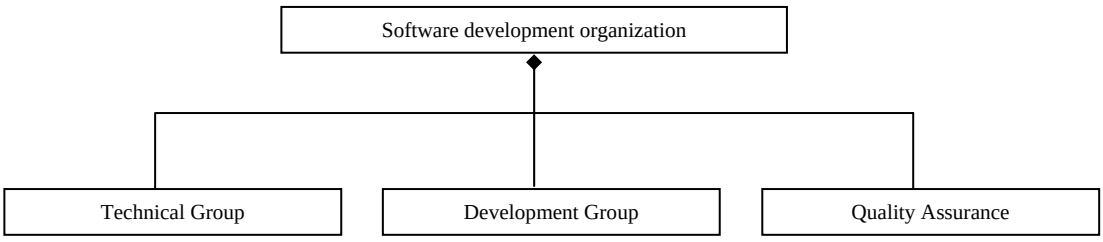


Figure 15. Class diagram of software development of XXX corp

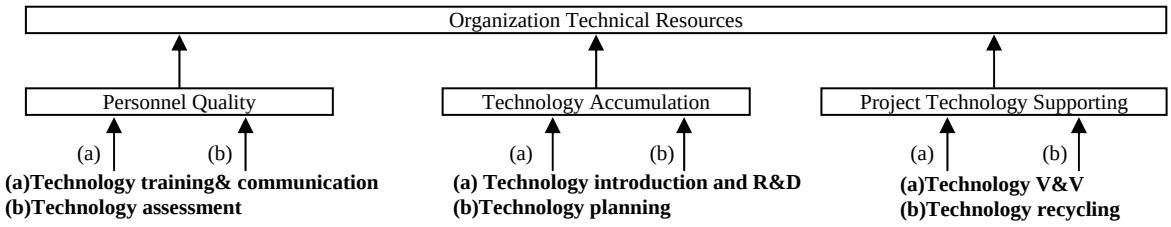


Figure 16. Structure diagram of software technology group of XXX corp

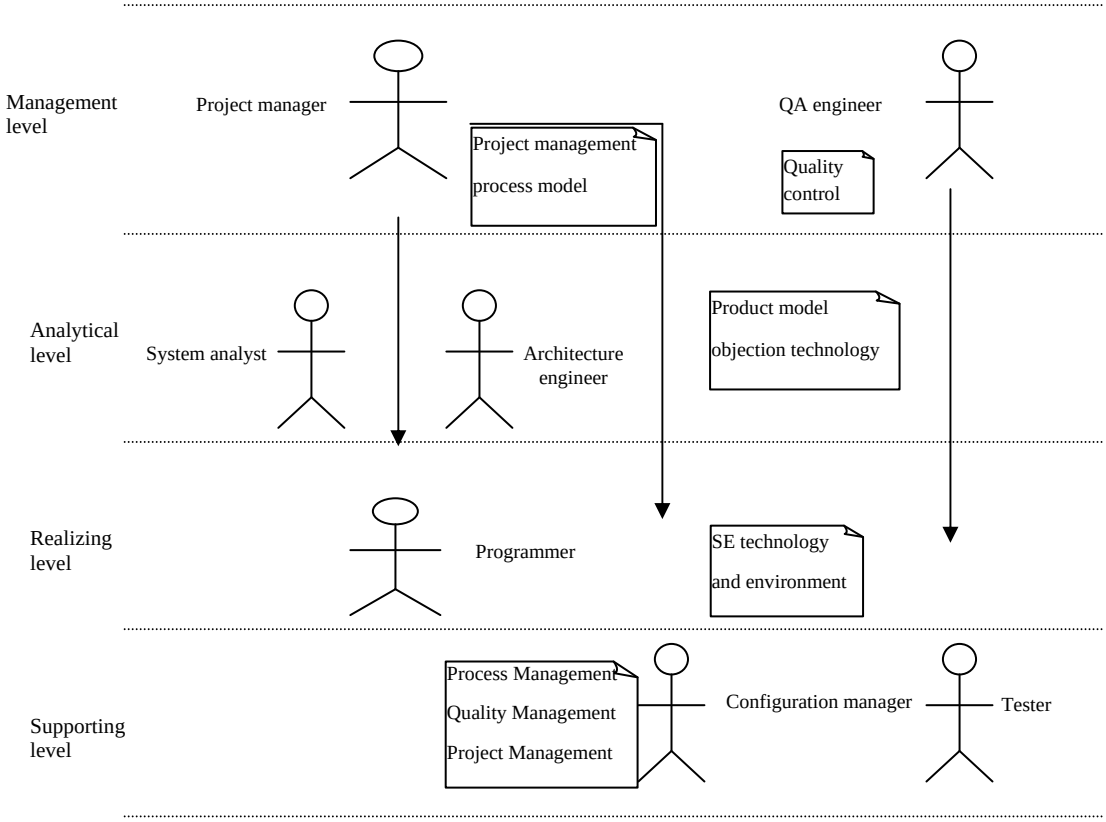


Figure 17. Software project organization of XXX corp

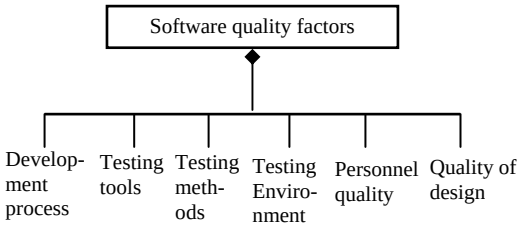


Figure 18. Software quality factors of XXX corp

9.2.3.2 Development Team

Development team can be further divided into a number of different conventional technology groups, such as network technology, Java technology, VB technology and PB technology. When the specific project is to be carried out, the project manager arranges temporary project teams according to the technical requirements. The organization of the project team is very important, so it should be decided by specifically analyzing the scale and technical characteristics of the project. The project organization is shown in Figure 17 (using the object-oriented technology, emphasizing on software system architecture, integrating the process model, product model and the object model, and describing that SPI needs project management knowledge, quality management expertise and software engineering knowledge and technology). If necessary, a project director can be set up to be responsible for total project control, cross-organizational coordination and policy management in order to achieve the project management and process management themes of the software process performance model.

9.2.3.3 Quality Assurance Group

The specific objectives of quality vary from different projects. It should be balanced between the investment in quality assurance and quality losses. Furthermore, it is important to keep the concept of quality and cost and customer satisfaction in mind so as to conduct the quality assurance in proper direction and eliminate the problems in the early phase. It is necessary to distinguish between products and customized projects in the quality requirements, and in the customized projects, the quality assurance can be agreed to be achieved by the cooperation with clients. Quality assurance activities include the quality planning, test plan, test analysis reporting, testing activities, proposal verification and process assessment and so on.

Many factors affect the quality, mainly in terms of the quality of development process, testing tools, testing methods, the quality of testing personnel, testing environment and quality of design (Figure 18). Through an

organic integration of processes, technology and personnel to achieve the process management theme of the software process performance model.

9.2.4 Keys to Software Project Management

9.2.4.1 Make Full Use of Available Resources

Development team is only one of the development executive team, the development standards and protocols, the power of the quality test, the accumulation reusable resources of enterprise (components, technology, methods, document templates, analogous case, business experts and the experience of members, etc.), are off-the-shelf resources that the development team can use, the project managers must make better use of these resources, especially good at excavating hidden resources, so as to shorten the development cycle.

9.2.4.2 Strengthen the Planning

Some people think that "program can't keep up with changes", so they hold a negative attitude to plans. But as long as we emphasis on planning and strengthen the management of a variety of changes, we can get two major benefits:

1) To reduce uncertainties. When people prepare plans, it is usually difficult to plan some problems. In fact, the uncertainties are on the exposed, we only need to assess the potential consequences of these uncertainties, and then formulate corresponding contingency measures. When people form a new version of each plans, the earlier uncertainties will be less. The evolution versions of plans are in fact clear step by step processes to the target.

2) Help to improve efficiency. Plans enhance a sense of urgency to reduce the waiting time for each other. Project team members are in a project, the plans make members all identify with the target, the process of formulating plans are actually the process of members' understanding, also is to clarify their own roles, responsibilities and the process of tasks. It is conducive to get members' psychological together, do what they should do, thereby enhancing the efficiency of the entire team.

9.2.4.3 Control Project Changes

We must control the project's direction of change toward favorable conditions, prevent adverse changes and eliminate a vicious circle. According to the control stage, control changes can be divided into three types:

1) Prior to control. The crucial issues must be ascertained in order to avoid opening the Pandora's Box. For example, the validity and the requirement-bound of the project contract documents must be confirmed, and the relevant documents should be signed in time in order to prevent a lot of projects change from time to time later because of people or time.

2) Control when it is in progress. For a variety of things appearing in reasonable changes, to format a legitimate change records after the deliberations of the relevant aspects, and have the same effect as the original legal documents. These records will serve as the basis for other related changes and later verification.

3) After control. In the circumstances that issues have emerged, we should properly handle them to prevent further deterioration of the problem, from big to small, small to little. At the conclusion of the project, projects changes arising in the course must be analyzed to guide future project planning and project control. Let members learn from the changes and improve the team's ability to cope with the changes.

9.2.4.4 Strengthen Requirement Management

Large applications need business experts, technical experts and other key personnel, to spend a great deal of energy to find it out, so it absolutely can't be ignored. The failure of most projects is not to find clear requirement. The purpose of requirement analysis is to ascertain needs, recognize the requirement and define boundaries, as well as to reduce rework due to requirement uncertainty. The following are requirement access steps:

1) Select staff. It is important for the entire quality assurance to choose the appropriate staff. Research and analysis staff needs to be better with rich experience and professional knowledge. Because the customer's requirement is often not clear, it is necessary to deal with requirement as a separate item as much as possible.

2) Research. Due to business customers often do not know how to tie in with the software development research, research methodology is very critical. We must grasp the whole, understand the details and then take the line of top-down. From the horizontal view, staff must research in accordance with the business processes from downstream to upstream backward (output and then import). It would be better to access needs from variety ways, such as form, face-to-face communication etc.

3) Writing. Write the requirement specifications.

4) Confirm. Recognizing the requirement, the two sides sign and generate an effective basis.

5) Change management. When management needs change, people must fill the requirement change list. It must be confirmed and then formatted the requirement tracking specifications.

9.2.4.5 Communication Management

Communication targets: project internal members, supervisors, customers, suppliers and supervisory company and so on. The regular medium of internal communications often is report forms and conference.

Communication forms: e-mail, fax, telephone and face-to-face and so on.

Five kinds of meeting: the project assessment conference, the demand review conference, the design review conference, and the product release conference.

To achieve the desired effect of communication, use e-mail or telephone and other forms in order to improve

the efficiency of communications as much as possible. For common resolutions and the content that must be addressed as an important basis of the communication, it is necessary to use fax and other written forms in order to improve the quality of communication. Written form communication is also a good way to reduce the communication misunderstanding, while oral communication is the most unreliable, which is mainly used to discuss, and its results ought to be recorded. The whole project written formal documents must be unified managed and archived collectively, having unified entrance and exit. Projects communications also need the right level and confidentiality rules.

9.2.4.6 Document Strategy

In accordance with the classification of software engineering methods, the development document can be divided into feasibility studies and planning category, requirements analysis category, outline design category, detailed design category, construction category, assembly and testing category, validation testing category, operation and maintenance category and so on. If there are too many documents, most people just cope with them in haste and pay no attention to the quality of documentation. Because a wide range of documents will increase the project cost, therefore, how to improve the efficiency to write documents is very important. The ways to enhance the management of documents are in the following:

Normative. We should establish a unified document language specification, edit, audit, grant, release, change rules. Build a complete documents list so that project staff can easily locate the right one.

Learn easily. We should establish document written guidelines and allow developers to share the document template, which can enhance the document's learn easily.

Readability. We should use charts, from the coarse-to-fine, surface to inner. Improve the document granularity. Facilitate supervision, accreditation and quality assurance personnel to participate. Write the document in accordance with the document and the reader (the development, use, training, implementation, sales, service and testing services), occupational characteristics and professional characteristics to reduce misunderstanding and the hardship to be understood.

Portability. We should pay attention to the document structure to facilitate the transplantation to other projects.

Tool support. We should use documentation tools (such as: UML tools, etc.) to automatically generate documentation.

Continuous improvement. We should provide the staff who writes documents with the continuous improving methods of document written.

9.2.4.7 Knowledge Management

The goal of knowledge management: Knowledge is the most valuable wealth in software companies. The stock, flow and network of knowledge reflect the accumulation and circulation of knowledge. Knowledge management is to collect the existing knowledge and skills in enterprise, and then to send them where it is needed and help enterprises maximize the benefits. The goal is to transfer the most appropriate knowledge in the most appropriate time to those most in need and help them make decision-making.

Table 6. The content of knowledge management

Classify	Item Details
Interior knowledge	1. The technical standards and management standards 2. Case-base 3. Questions-base 4. Knowledge-base 5.The public library module 6.The technical exchange activities 7.The registration of intellectual property rights (copyright registration)
Exterior knowledge	1. Establish the feedback information base to provide the product development requirements for new technologies 2. The supplier information database: For example, training institutions, the servers, match software, etc. 3. Competitor information base 4. The expert knowledge base (explain how to obtain the means of knowledge if it doesn't have electronic document) 5. Outsourced staff expertise 6. The latest technology terms 7. The relevant knowledge Web address
Staff knowledge creation	1. Staff expertise library, that can be easily convened to discuss technical problems 2. Staff recommends library 3. Troubleshooting

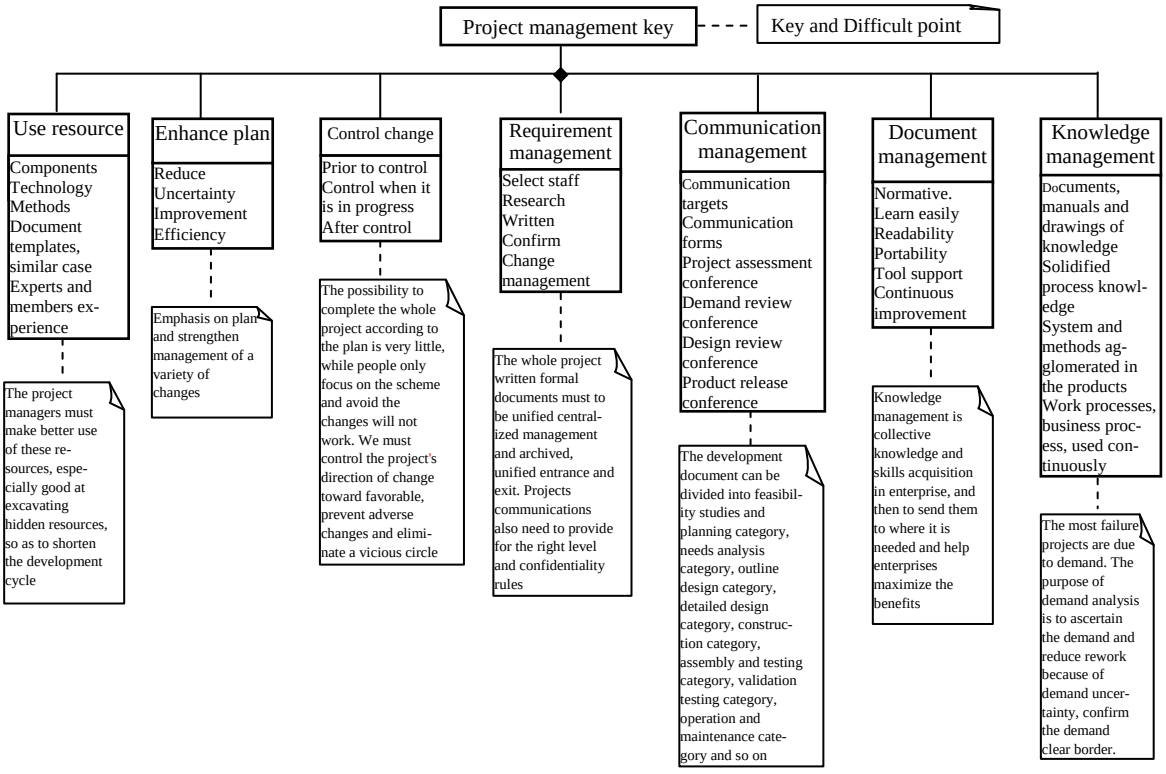


Figure 19. Class diagram of keys to software project management of XXX corp

The existing forms of knowledge: archives (documents, manuals and drawings of knowledge), the tacit knowledge in human brain, solidified process knowledge (system and methods agglomerated in the products, work processes, business processes, which are used continuously).

Knowledge management tasks: To build the knowledge store, improve the knowledge acquisition methods and the knowledge environment, manage knowledge assets. Set up knowledge management positions to collect and organize knowledge. Establish knowledge networks to provide the access for the project team at any time. Brainstorming, take full advantage of the tacit knowledge in organizations.

We should pay attention to sum up the project, analyze and extract useful knowledge archive. Table 6 is the classification and illustration of knowledge management that contribute to knowledge management. Figure 19 shows the key issues and points to implement software project management.

9.2.5 Knowledge Integration Support Structure of Quality Software Production

9.2.5.1 Japanese Knowledge Science

Ikurjiro Nonaka argued that knowledge is created and used in organization through knowledge transforming process, including four patterns: socialization, externalization, combination, and internalization. All four of these patterns exist in dynamic interaction, a kind of spiral of knowledge. Knowledge is truly created and used effectively and depending on established dynamic business system [22–23]. Nine questions are put forward to the following [23]: 1) What is the status of “truth” in the definition of knowledge? 2) Do tacit and explicit knowledge fall along a continuum? 3) Is the tacit/explicit knowledge distinction along the continuum valuable for organization science? 4) What is the conceptual basis of knowledge conversion? 5) Given the relationship between tacit knowledge and social practices, how can the concept of knowledge conversion be upheld? 6) What is the outcome of knowledge conversion? 7) What is the relationship between organizational knowledge creation and social practices in organizations? 8) When and why do social practices contribute to the conservation of existing tacit knowledge and existing routine rather than organizational knowledge creation and innovation? 9) How can leadership motivate and enable individuals to contribute to organizational knowledge creation by transcending social practices? College of Know- ledge Sciences was established in Japan Advanced Institute of Science and Technology (JAIST) to investigate and study how to establish and develop knowledge science in the world [24].

9.2.5.2 Chinese Knowledge Engineering

Prof. Wang Zhongtuo defined the knowledge technology as the following: knowledge technology is the discipline about the methodologies, methods, procedures and tools of knowledge identification, acquisition, storage, processing, dissemination, and creation for the purpose of adding value to human economic and social activities. The object of knowledge technology is not only the explicit knowledge, but also tacit knowledge and their interrelations. The approaches are not only technique-oriented, but also human-behavior-oriented. The knowledge management involves primarily the knowledge processing cycle that includes the capture, analysis and communication of knowledge within organization. It also involves problems of searching and finding useful information. But the most important thing is the creation of new knowledge. He suggested that knowledge system engineering is the discipline of organization and management of knowledge systems. The architecture of knowledge systems, operation process of knowledge systems, engineering project development, systems intuition and knowledge fusion are under studying. Research center of knowledge science and technology was established in Dalian University of Technology (DUT) in 2000 [24].

9.2.6 The Domain of Science Model and the Domain of Knowledge Model

9.2.6.1 The Domain of Science Model

Figure 20 illustrated the domain of science model by Prof. Warfield in 1986.The two blocks consist of the corpus of the science. The two blocks consisting of the methodology and applications make up what is called arean. The arean is, of course, where the action is, which is often identified with "action research": a kind of research basing on the view that there is no substitute for becoming oneself in the problem as it occurs in a particular situation. The corpus refers to a body of knowledge but not to application-specific knowledge. The domain of science model is initially divided to reflect four components or blocks. There are: foundations, theory, methodology, and applications. The first three of these make up the science. The model represents a relationship of steering in that the foundations steer the theory, the theory steers methodology, and methodology steers applications (and through the foundations the other blocks of the science) [4].

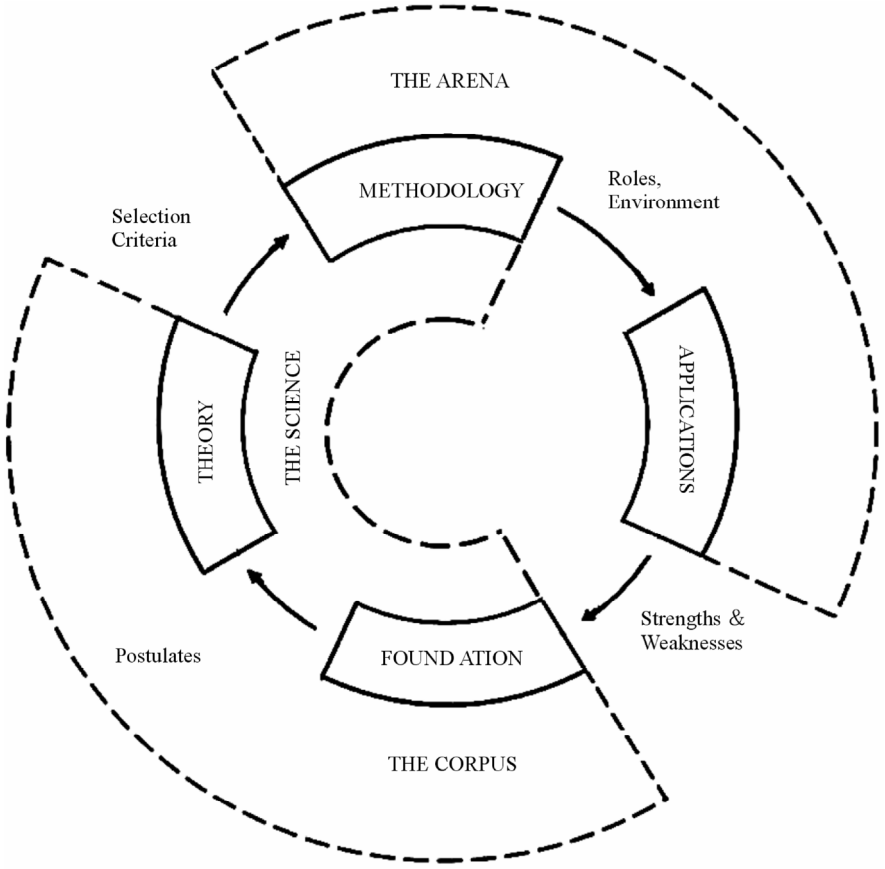


Figure 20. The domain of science model

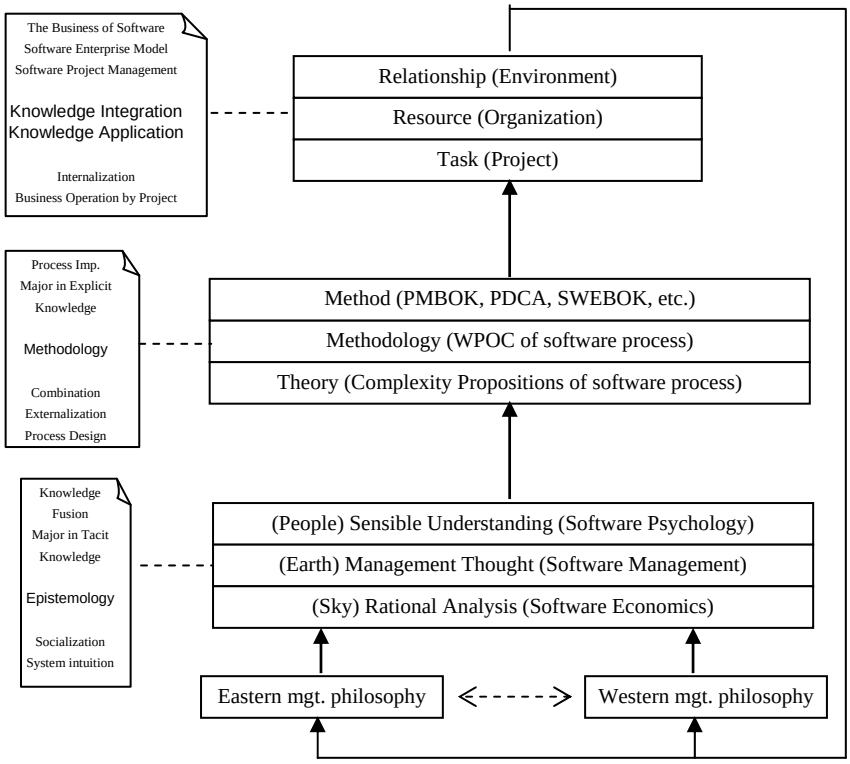


Figure 21. Knowledge integration support structure of quality software production

The foundations in the model have the specific function of providing the decision-making basis for the science. Whenever issues arise in the theory or the methodology, it must be possible to refer them to the foundations for resolution. If it is not possible to resolve an issue in this way, one must assume that the foundations are, in some way, lacking and must be upgraded. In order to fulfill its function, the foundations must incorporate the Universal Priors (the human being, language, reasoning through relationships, and archival representations) in a way that is directly relevant to the particular science being constructed. The foundation also must incorporate those particular concepts (the essences) that are required to distinguish the particular science from other sciences [4].

In order for the foundations to steer the theory, it is clear that the foundations must be prior to the theory, i.e., they must contain concepts and propositions that do not depend on still deeper ideas foreign to the science and they must not depend on the theory. The dependence must be from the foundations to theory. A certain parsimony is necessary; but matters must be excluded that are not so established. To fulfill the function of being a fount of resolution for issues arising in the Science, the Foundations must not too great in number. To the extent possible, ideas should move out of the foundations toward the theory and out of the theory toward the methodology, where greater volumes of information can be tolerated [4].

If the various sciences were clearly organized in terms of the three blocks of the Domain of Science Model, people who have to work across the sciences would find the task of drawing upon them much easier. And this would permit the integration of parts of the recognized scientific disciplines into newer integrative sciences [4].

9.2.6.2 The Domain of Knowledge Model

In our understanding, philosophy beliefs, which steer what type of theory being established by knowledge workers, must be involved. The “methodology” originally refers route in which one traces another and it evolves the principles and program to do work. Methods must divorce from methodologies and becomes new layer. In addition, the foundation of a given discipline can be merged with its relative theory layer. The domain of knowledge science is established referring the domain of science model, including philosophy belief, theory, methodology, methods and applications. Of course, the applications are also counteractive on the philosophy belief. Because the philosophy does not belong to the science category, it is referred to as the domain of knowledge model [10].

9.2.7 Knowledge Integration Support Structure of Quality Software Production

Software process is a collection of complex activities that have strict time sequences, some are currently asynchro-

nous, and some are condition each other and some interact with each other. In fact, the activities in software process have complex network relationship. Software development are highly dynamic processes, and the dynamic change can be found in all phases in software processes, such as requirement specification, assignment tasks, debugging, development policy, tools and support environment etc. These changes are often unpredictable and their influences often couldn't be identified because software process is people-oriented and software product is intangible itself, and this results in that the software processes are hard to control and theirs work quality are not easy to assess. Software process has great complexity. What philosophy beliefs are necessary for knowledge economy? How to design methodology of quality software production management? How to align quality software production management with software enterprise business objects to compete in both domestic market and overseas, which are rapidly changing?

Basing on both the domain of knowledge model and support structure of interactive management, knowledge integration support structure of quality software production is illustrated in Figure 21.

Figure 21 illustrates that managing the complexity of quality software production can be abstract through essence (e.g. the complexity propositions of knowledge management of software process etc.), and the abstract problems can be embodied through interactive cognition process (process design, e.g. work program of complexity of knowledge management of software process etc.), and the embodied problems to be structured through learning process continuously (process implementation, e.g. PMBOK, PDCA and SWEBOK etc.), and the structural problems to realize the business of software through business operation by project (e.g. knowledge integration model of software enterprise etc.).

The keys to knowledge integration support structure of quality software production may be knowledge fusion among Eastern management philosophy, Western management philosophy, sensible understanding (People, Ren-li and Software Psychology), management thought (Earth, Wu-li and Software Management) and rational analysis (Sky, Shi-li and Software Economics). In other words, tacit knowledge of software is socialized. At the same time, the business of software is also reaction on its support elements. “Sky-Earth-People” is the “Three-Cai” of Change of Science [25], which is origin of Chinese traditional philosophy, and “Shi-li Wu-li Ren-li” is put forward by Prof. Gu jifa and Prof. Zhu zhichang. “Software Psychology – Software Management – Software Economics” is adopted from software terms.

Prof. Cheng Zhongying argued that management always depends on a structure and evolves to an adaptable process [25]. In our understanding, quality software production depends on knowledge integration support structure which is illustrated in Figure 21, and evolves to design process with continuously transforming, creating and integrating process between explicit knowledge and tacit knowledge, and it's also an organizational learning process to resolve cognitive complexity continuously from system intuitions, and knowledge fusion, and process’s design and process's implementation, and business operation by project to the business of software. Because software industry is the core of knowledge economic, the knowledge integration support structure of quality software production is also generic resolution for managing complexity of knowledge systems.

10. Conclusions

Based on our understanding, there is a long way to go for knowledge integration support structure of quality software production in both theory and practice according to the domain of science model (Warfield), knowledge science (Ikujiro Nonaka) and knowledge technology (Wang Zhongtuo), but it will certain that the research is useful for both software production and knowledge economy in the future.

11. Acknowledgment

Thanks for the helpful discussion with Prof. Warfield, Prof. Wang Zhongtuo, Prof. Yang Jianmei, Mr. Li Jiangzhuang, Mr. Hou Yawen, Mr. Zhou Qiyang, Mr. Zhou Zhijun and my studtets Liu Qingjing, Wan Dan etc.

REFERENCES

[1] W. S. Humphrey, “Managing the software process,” Reading. MA: Addison-Wesley, pp. 19–24, 1989.

[2] J. D. Herbsleb and D. R. Goldenson, “A systematic survey of CMM experience and results,” in Proceedings 18th Intennational Conference on Software Engineering, Berlin, Germany, pp. 323–330, March 1996.

[3] R. S. Pressman, “Software engineering: A practitioner’s approach,” Vol. 5, McGraw-Hill Companies, Inc., pp. 19, 2001.

[4] J. N. Warfield, “A science of generic design: Managing complexity through systems design,” IOWA State University, pp. 142–146, 187–188, 1994.

[5] J. N. Warfield and A. R. Cardenas, “A handbook of interactive management,” AJAR Publishing Company, 1994.

[6] J. N. Warfield, “Twenty laws of complexity: Science applicable in organizations,” Systems Research and Behavioral Science, Vol. 16, No. 1, pp. 3–40, 1999.

[7] J. N. Warfield, “Understanding complexity: Thought and behavior,” AJAR Publishing Company, 2002.

[8] M. Scott and J. N. Warfield, “Enterprise integration of product development data: Systems science in action,”

Enterprise Information Systems, Vol. 1, No. 3, pp. 269–285, August 2007.

[9] Brooks and P. Frederick, "No silver bullet: Essence and accidents of software engineering," Computer, Vol. 20, pp. 10–19, April 1987.

[10] J. P. Wan and J. M. Yang, "Research on the work program of complexity of software process improvement—Methodology for implementation of SW-CMM," Science Press, Beijing, 2004. (in Chinese)

[11] J. P. Wan and J. M. Yang, "On the meanings of complexity, generic design science and work program of complexity," Journal of Systemic Dialectics, Vol. 10, No. 4, pp. 41–44, December 2002. (in Chinese)

[12] J. P. Wan and J. M. Yang, "Interaction management and its application," International Journal of Knowledge and Systems Science, Vol. 2, No. 1, pp.25–32, March 2005.

[13] J. P. Wan and J. M. Yang, "Knowledge management in software process improvement," Application Research of Computer, Vol. 19, No. 5, pp. 1–3, May 2002. (in Chinese)

[14] J. P. Wan and J. Z. Li, "Some considerations on knowledge management in software enterprise," Application Research of Computer, Vol. 20, No. 1, pp. 13–16, January 2003. (in Chinese)

[15] J. P. Wan and Y. L. Zhuo, "The e-business challenge," Application Research of Computer, Vol. 20, No. 9, pp. 9–10, September 2003. (in Chinese)

[16] J. Z. Li and J. P. Wan, "Considerations on project management in small and middle software organization," Application Research of Computer, Vol. 20, No. 9, pp. 14–17, September 2003. (in Chinese)

[17] J. P. Wan and J. M. Yang, "Research on the complexity of software process improvement," In Proceedings of 2003 International Conference on Management Science & Engineering, Moscow, USA, pp. 168–172, August 15-17, 2003.

[18] J. P. Wan, J. M. Yang, and H. Y. Han, "Support structure of knowledge management in software process improvement," In Information Systems: e-business Challenge, IFIP 17th World Computer Conference, Montreal, Canada, pp. 17–29, August 25-30, 2002.

[19] B. W. Boehm, "Seven basic principles of software engineering," Journal of Systems and Software, Vol. 3, pp. 3–24, March 1983.

[20] Z. Y. Zhou, "CMM in uncertain environments," Communication of the ACM, Vol. 46, No. 8, pp. 8–27, August 2004.

[21] J. N. Warfield, "Mentomology: The identification and classification of mindbugs," http://mars.gmu.edu:8080/dspace/bitstream/1920/3199/1/Warfield%20_20_20_A1b.pdf, 1995.

[22] Nonaka and Ikjurio, "Dynamic theory of organizational knowledge creation," Organization Science, Vol. 5, No. 1, pp. 14–36, 1994.

[23] Ikujiro Nonaka and Georg von Krogh, "Perspective—Tacit knowledge and knowledge conversion: Controversy and advancement in organizational knowledge creation theory," Organization Science, Vol. 20, No. 3, pp. 635–652, 2009.

[24] Z. T. Wang, Knowledge Systems Engineering, Beijing, Science Press, 2004. (in Chinese)

[25] Z. Y. Cheng and C. Theory, The Management Philosophy of China, Shanghai, XueLin Press, Vol. 315, 1999. (in Chinese)

J. Software Engineering & Applications, 2009, 2: 195-199
doi:10.4236/jsea.2009.23026 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Formal Derivation of the Combinatorics Problems with *PAR* Method

Lingyu SUN[1], Yatian SUN[2]

[1]Department of Computer Science, Jinggangshan University, Ji'an, China; [2]School of Chemistry and Materials Science, University of Science and Technology of China, Hefei, China.
Email: sunlingyu@jgsu.edu.cn

Received May 12th, 2009; revised June 13th, 2009; accepted June 17th, 2009.

ABSTRACT

Partition-and-Recur (PAR) method is a simple and useful formal method. It can be used to design and testify algorithmic programs. In this paper, we propose that PAR method is an effective formal method on solving combinatorics problems. Furthermore, we formally derive combinatorics problems by PAR method, which cannot only simplify the process of algorithmic program's designing, but also improve its automatization, standardization and correctness. We develop algorithms for two typical combinatorics problems, the number of string scheme and the number of error permutation scheme. Lastly, we obtain accurate C++ programs which are transformed by automatic transforming system of PAR platform.

Keywords*: *PAR Method, Formal Derivation, Combinatorics, Algorithmic Programs

1. Introduction

Computer Science is a science of developing correct and efficient algorithms. Its main research object is discrete data handled by computer. As long as the algorithm involves iterant calculation, each traditional strategy of algorithm design can embody the principle of partition and recurrence. Partition is a general approach for dealing with complicated objects and is typically used in divide-and-conquer approach. Recurrence is used in algorithm analysis and dynamic programming approach. *PAR* method is a unified approach of developing efficient algorithmic programs and its key idea is partition and recurrence. Using *PAR* method, we begin from the formal specification of a specified problem, partition the problem into a couple of sub problems, and develop an efficient and correct algorithm represented by recurrence and initiation [1−3].

Combinatorics is a science of studying discrete objects. In general, it includes basic theories, counting methods and algorithms widely used in combination. Combinatorics algorithms are based on combinatorics and make people feel that computer may have its own thought.

However, many programming teaching materials can only present combinatorics algorithms but cannot provide the formal derived procedures that design the specific algorithms from the unresolved combinatorics problems. So algorithm designers cannot understand the essence of algorithms and cannot improve their capability of algorithm design [4].

PAR method embodies rich mathematic ideology, it provides many powerful tools and appropriate developing environment. We can avoid making a choice among various design methods by adopting *PAR* method to develop combinatory algorithms. It can also change many creative labor to mechanized labor, and can finally improve algorithmic programs design's automatization, standardization and correctness [5,6].

In this paper, we propose that *PAR* method is an effective formal method on solving combinatorics problems. We formally derive several typical algorithms of combinatorics problems by *PAR* method, which cannot only solve the above problems, but also can assure their correctness on logic. These combinatorics problem instances include the number of string scheme, the number of error permutation scheme, maximum summary [4], the longest common subsequence [4], minimum spanning tree [4,7] and the Knapsack problem [4], etc. Because of the limited space, we only describe the whole developing process of two specific combinatorics problem instances, the number of string scheme and the number of error permutation scheme, which are derived by *PAR* methods.

2. The Developing Steps of the Number of String Scheme Derived by *PAR* Methods

Suppose that A=$\{a,b,c,d\}$, n is given, and we want to

select n elements to compose string, in which element a and b cannot be adjacent elements. How many schemes are there in a string with n elements? [4]

2.1 The Formal Function Specification of the Number of String Scheme

PQ: Given integer n, set $A=\{a,b,c,d\}$, string S stores n elements of set A.

PR: Let $String(n)=\{S|S[k]=a\vee S[k]=b\vee S[k]=c\vee S[k]=d\wedge 1\le k<n\}$ and $F(S,i)=\forall(j:i\le j<n:S[j..j+1]\neq ab\wedge S[j..j+1]\neq ba)$, then

$$\begin{aligned}Z_n &= N(S:S\in String(n):\\ &\quad \forall(j:1\le j<n:S[j\ldots j+1]\neq ab\wedge S[j\ldots j+1]\neq ba))\\ &= \{\textit{According to the definition of } F(S,i)\}\\ &\quad N(S:S\in String(n):F(S,1))\\ &= \{\textit{equivalent transformation of quantifier}\}\\ &\quad \sum(S:S\in String(n)\wedge F(S,1):1)\end{aligned}\tag{1}$$

2.2 Partition the Problem Based on the Post-Assertion

We partition computing Z_n into computing X_n and Y_n, each of that has the same structure with Z_n.

$$\begin{aligned}X_n &= \sum(S:S\in String(n)\wedge F(S,1)\\ &\quad \wedge(S[1]=a\vee S[1]=b):1)\end{aligned}\tag{2}$$

In Equation (2), X_n denote the number of n elements' string scheme that element a and b cannot be adjacent elements and the initial element is a or b.

$$\begin{aligned}Y_n &= \sum(S:S\in String(n)\wedge F(S,1)\\ &\quad \wedge(S[1]=c\vee S[1]=d):1)\end{aligned}\tag{3}$$

In Equation (3), Y_n denote the number of n elements' string scheme that element a and b cannot be adjacent elements and the initial element is c or d.

$$\begin{aligned}Z_n &= \sum(S:S\in String(n)\wedge F(S,1):1)\\ &= \sum(S:S\in String(n)\wedge F(S,1)\\ &\quad \wedge(S[1]=a\vee S[1]=b\vee S[1]=c\vee S[1]=d):1)\\ &= \{\textit{Range Disjunction}\}\\ &\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=a\vee S[1]=b):1)+\\ &\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=c\vee S[1]=d):1)\\ &= \{\textit{According to the definition of } X_n \textit{ and } Y_n\}\\ &\quad X_n+Y_n\end{aligned}\tag{4}$$

According to the equation (4), we can partition computing Z_n into computing X_n and Y_n.

2.3 Construct the Recurrence Relation

Suppose X_{n-1} denotes the number of *n-1* elements' string scheme that satisfy the condition and the initial element is a or b. Y_{n-1} denotes the number of *n-1* elements' string scheme that satisfy the condition and the initial element is c or d.

$$\begin{aligned}X_n &= \sum(S:S\in String(n)\wedge F(S,1)\\ &\quad \wedge(S[1]=a\vee S[1]=b):1)\\ &= \sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]\\ &= a\vee S[1]=b)\wedge(S[2]\\ &= a\vee S[2]=b\vee S[2]=c\vee S[2]=d):1)\\ &= \{\textit{Range Disjunction}\}\\ &\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=S[2])\\ &\wedge(S[2]=a\vee S[2]=b):1)+\\ &\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=a)\\ &\wedge(S[2]=c\vee S[2]=d):1)+\\ &\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=b)\\ &\wedge(S[2]=c\vee S[2]=d):1)\\ &= \{\textit{According to the definition of } F(S,i)\}\\ &\sum(S:S\in String(n)\wedge(S[1]=S[2])\wedge F(S,2)\\ &\wedge(S[2]=a\vee S[2]=b):1)+\\ &\sum(S:S\in String(n)\wedge(S[1]=a)\wedge F(S,2)\\ &\wedge(S[2]=c\vee S[2]=d):1)+\\ &\sum(S:S\in String(n)\wedge(S[1]=b)\wedge F(S,2)\\ &\wedge(S[2]=c\vee S[2]=d):1)\\ &= \{\textit{According to the definition of } X_{n-1} \textit{ and } Y_{n-1}\}\\ &\quad X_{n-1}+2\times Y_{n-1}\end{aligned}\tag{5}$$

According to the equation (5), we can partition computing X_n into computing X_{n-1} and $2\times Y_{n-1}$.

$$\begin{aligned}Y_n &= \sum(S:S\in String(n)\wedge F(S,1)\\ &\quad \wedge(S[1]=c\vee S[1]=d):1)\\ &= \sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]\\ &= c\vee S[1]=d)\wedge(S[2]=a\vee S[2]\\ &= b\vee S[2]=c\vee S[2]=d):1)\end{aligned}$$

$= \{Range\ Disjunction\}$

$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=c\vee S[2]=d)\wedge(S[2]=a\vee S[2]=b):1)+$$
$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=c\vee S[2]=d)\wedge(S[2]=c\vee S[2]=d):1)$$

$= \{Range\ Disjunction\}$

$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=c)\wedge(S[2]=a\vee S[2]=b):1)+$$
$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=d)\wedge(S[2]=a\vee S[2]=b):1)+$$
$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=c)\wedge(S[2]=c\vee S[2]=d):1)+$$
$$\sum(S:S\in String(n)\wedge F(S,1)\wedge(S[1]=d)\wedge(S[2]=c\vee S[2]=d):1)$$

$= \{According\ to\ the\ definition\ of\ \ F(S,i)\}$

$$\sum(S:S\in String(n)\wedge(S[1]=c)\wedge F(S,2)\wedge(S[2]=a\vee S[2]=b):1)+$$
$$\sum(S:S\in String(n)\wedge(S[1]=d)\wedge F(S,2)\wedge(S[2]=a\vee S[2]=b):1)+$$
$$\sum(S:S\in String(n)\wedge(S[1]=c)\wedge F(S,2)\wedge(S[2]=c\vee S[2]=d):1)+$$
$$\sum(S:S\in String(n)\wedge(S[1]=d)\wedge F(S,2)\wedge(S[2]=c\vee S[2]=d):1)$$

$= \{According\ to\ the\ definition\ of\ X_{n-1}\ and\ Y_{n-1}\}$

$$2\times X_{n-1}+2\times Y_{n-1} \quad (6)$$

According to the equation (6), we can also partition computing Y_n into computing $2\times X_{n-1}$ and $2\times Y_{n-1}$.

2.4 Developing Loop Invariant

Suppose variant x stores the value of X_i, variant u stores the value of X_{i+1}, variant y stores the value of Y_i, variant v stores the value of Y_{i+1}, variant z stores the value of Z_i, where $X_1=2$, $Y_1=2$, $Z_1=4$.

LI: $x=X_i\wedge u=X_{i+1}\wedge y=Y_i\wedge v=Y_{i+1}\wedge z=Z_i\wedge(1\le i\le n)$

2.5 Developing Corresponding *RADL* Program

The Recurrence-based Algorithm Design Language (*RADL*) program of the number of string scheme, derived by *PAR* methods, is shown in Algorithm 1. By the automatic program transforming system of *PAR* platform, we can get the Abstract Programming Language (*APLA*) program of the number of string scheme which is transformed from the *RADL* program and is shown in Algorithm 2. Finally, we transform the *APLA* program of the number of string scheme to *C++* program, which can get accurate running result.

Algorithm 1 (*The RADL program of string scheme*)

|[**in** *n:integer; i:integer; x,y,u,v: integer;* **out** *z:integer;*]|

{**PQ**∧**PR**}

Begin: i=1++1; x=2; y=2; z=4;

A_I: $x=x(i)\wedge u=x(i+1)\wedge y=y(i)\wedge v=y(i+1)\wedge z=z(i)\wedge(1\le i\le n)$

Termination: i=n;

Recur: u=x+2*y; v=2*x+2*y;
z=u+v; x=u; y=v;

End.

Algorithm 2 (*The APLA program of string scheme*)

```
var: n:integer; i:integer; x,y,u,v: integer; z:integer;
begin:
   write("Please input integer n value");
   read(n);
   i:=1;x:=2;y:=2;z:=4;
   do (¬(i=n)) →
         u:=x+2*y;
         v:=2*x+2*y;
         z:=u+v
         x:=u;
         y:=v;
         i:=i+1;
   od
   write("The Number of string scheme:", z);
end.
```

3. The Developing Steps of the Number of Error Permutation Scheme Derived by *PAR* Methods

Suppose that the original arrange scheme is $A=[a_1\cdots a_i\cdots a_n]$, which is satisfied with each element is different. We want to interlace n elements of original permutation scheme A to compose new permutation scheme B, in which each element cannot be the same position in A. How many schemes are there in error permutation with n elements? [4]

3.1 The Formal Function Specification of the Number of Error Permutation Scheme

PQ: Given the original permutation scheme $A=[a_1\cdots a_i\cdots a_n]$, which is satisfied with $\forall(i,j:$

$(1 \le i < j \le n) : (A[i] \ne A[j]))$.

PR: Let $Perm(A) = \{B \mid B[j] = A[i] \wedge B[j] \ne B[k] \wedge 1 \le i, j, k \le n \wedge j \ne k\}$, then

$$D_n = N\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right)\right)$$

= {*equivalent transformation of quantifier*}

$$\sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) : 1\right)$$

$$= \{\exists'(i : 1 \le i < n : B[n] = A[i]) = true\}$$

$$\sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i \le n : B[n] = A[i]\right) : 1\right) \quad (7)$$

3.2 Partition the Problem Based on the Post-Assertion

We partition computing D_n into computing U_n and V_n, each of that has the same structure with D_n.

$$U_n = \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] = A[n]\right) : 1\right) \quad (8)$$

In Equation (8), U_n denotes the number of n elements' error permutation scheme in which a_n must be the i^{th} element in new permutation B.

$$V_n = \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] \ne A[n]\right) : 1\right) \quad (9)$$

In Equation (9), V_n denotes the number of n elements' error permutation scheme in which a_n cannot be the i^{th} element in new permutation B.

$$D_n = \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i]\right) : 1\right)$$

= {*Range Disjunction*}

$$\sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] = A[n]\right) : 1\right) +$$

$$\sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] \ne A[n]\right) : 1\right)$$

= {*According to the definition of U_n and V_n*}

$$U_n + V_n \quad (10)$$

According to the equation (10), we can partition computing D_n into computing U_n and V_n.

3.3 Construct the Recurrence Relation

Suppose D_{n-1} denotes the number of *n-1* elements' error permutation scheme, D_{n-2} denotes the number of *n-2* elements' error permutation scheme.

$$U_n = \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] = A[n]\right) : 1\right)$$

= {*Generalized Range Disjunction*}

$$\sum\left(i : 1 \le i < n : \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge B[n] = A[i] \wedge B[i] = A[n] : 1\right)\right)$$

$$= \sum\left(i : 1 \le i < n : \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j < i \vee i < j \le n : B[j] \ne A[j]\right) : 1\right)\right)$$

= {*According to the initial definition of D_n*}

$$\sum\left(i : 1 \le i < n : D_{n-2}\right) = (n-1) \times D_{n-2} \quad (11)$$

According to the equation (11), we can partition computing U_n into computing $(n\text{-}1) \times D_{n-2}$.

$$V_n = \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge \exists'\left(i : 1 \le i < n : B[n] = A[i] \wedge B[i] \ne A[n]\right) : 1\right)$$

= {*Generalized Range Disjunction*}

$$\sum\left(i : 1 \le i < n : \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j \le n : B[j] \ne A[j]\right) \wedge B[n] = A[i] \wedge B[i] \ne A[n] : 1\right)\right)$$

$$= \sum\left(i : 1 \le i < n : \sum\left(B : B \in Perm(A) \wedge \forall\left(j : 1 \le j < n : B[j] \ne A[j]\right) : 1\right)\right)$$

= {*According to the initial definition of D_n*}

$$\sum\left(i : 1 \le i < n : D_{n-1}\right) = (n-1) \times D_{n-1} \quad (12)$$

According to the equation (12), we can partition computing V_n into computing $(n\text{-}1) \times D_{n-1}$.

According to the equation (10), (11), (12), we have the recurrence: $D_n = U_n + V_n = (n\text{-}1) \times (D_{n-1} + D_{n-2})$, where $D_1 = 0$, $D_2 = 1$. That is to say, we can partition computing D_n into computing $(n\text{-}1) \times D_{n-1}$ and $(n\text{-}1) \times D_{n-2}$.

3.4 Developing Loop Invariant

Suppose variant *s1* stores the value of *d*(*i-2*), variant *s2*

stores the value of $d(i\text{-}1)$, variant d stores the value of $d(i)$, where D_1 =0, D_2=1.

LI: $s1 = d(i-2) \wedge s2 = d(i-1) \wedge d = d(i)$ $\wedge(1 \le i \le n)$

3.5 Developing Corresponding *RADL* Program

The *RADL* program of the number of error permutation scheme, derived by the *PAR* methods, is shown in Algorithm 3. By the automatic program transforming system of *PAR* platform, we can get the *APLA* program of the number of error permutation scheme which is transformed from the *RADL* program and is shown in Algorithm 4. Finally, we transform the *APLA* program of the number of error permutation scheme to *C++* program, which can get accurate running result.

Algorithm 3 (*The RADL program of error permutation scheme*)

|[in *n:integer; i,s1,s2: integer*; out *d:integer*;]|
{**PQ**∧**PR**}
Begin: i=3++1; s1=0; s2=1;
A_I: $s1 = d(i-2) \wedge s2 = d(i-1) \wedge d = d(i)$ $\wedge(3 \le i \le n)$
Termination: i=n+1;
Recur: d= (i-1)*(s1+s2);
s1=s2;s2=d;
End.

Algorithm 4 (*The APLA program of error permutation scheme*)

```
var: n:integer; i:integer; s1,s2: integer; d:integer;
begin:
   write("Please input integer n value");
   read(n);
   i:=2;s1:=0;s2:=1;
   do (¬(i=n)) →
         d= (i-1)*(s1+s2);
         s1:=s2;
         s2:=d;
         i:=i+1;
   od
   write("The Interlaced Arrange Scheme:", z);
end.
```

4. Conclusions

We developed algorithmic programs for the number of string scheme and the number of error permutation scheme which are typical combinatorics problems. It is revealed that PAR method has particular merit. It is apt to understand and demonstrate the ingenuity and correctness of an algorithm by formula deduction. Compared with other derivation, our algorithmic programs are more precise and simple than the representation of algorithm in natural language, flowchart and program. It also shows that using *PAR* method in developing combinatorics problem is a very natural meaningful research work. This can not only expand the application of *PAR* method, but also prove that the *PAR* method is a unified and effective approach on solving combinatorics problems.

5. Acknowledgment

This paper is supported by the international cooperation project of Ministry of Science and Technology of PR China, grant No. CB 7-2-01, and by Science and Technology research project of Jiangxi Municipal Education Commission under Grant No. GJJ09590.

REFERENCES

[1] J. Y. Xue, "A unified approach for developing efficient algorithmic programs [J]," Journal of computer Science and Technology, Vol. 12, No. 4, pp. 103–118, 1997.

[2] J. Y. Xue, "Formal derivation of graph algorithmic programs using Partition-and-Recur [J]," Journal of Computer Sciences and Technology, Vol. 13, No. 6, pp. 95–102, 1998.

[3] J. Y. Xue, "A practicable approach for formal development of algorithmic programs [C]," The Proceedings of The international Symposium on Future software Technology (ISFS'99), Published by Software Engineers Associations of Japan, pp. 212–217, 1999.

[4] L. Y. Sun, "The applied research of PAR method on combinatorics problems [D]," Jiangxi Normal University, 2007.

[5] J. Y. Xue, "Research on formal development of algorithmic program [J]," Journal of Yunnan University (natural sciences), Vol. 19, pp. 283–288, 1997.

[6] Y. Q. Li, "Partition-and-recur method and its applications [J]," Computer Engineering and Applications, Vo1. 27, No. 11, pp. 77–79, 2000.

[7] L. Y. Sun and J. Y. Xue, "Formal derivation of the minimum spanning tree algorithm with PAR Method [J]," Computer Engineering, Vo1. 32, No. 21, pp. 85–87, 2007.

J. Software Engineering & Applications, 2009, 2: 200-205
doi:10.4236/jsea.2009.23027 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Sharing and Implementation of Heterogeneous Database for Education Resource Based on XML

Shixi TANG

YanCheng Teachers University College of Science & Technology, Yancheng, China.
Email: tsxlyh@163.com

Received May 11th, 2009; revised July 1st, 2009; accepted July 15th, 2009.

ABSTRACT

The problem of sharing heterogeneous database for accessing different educational resources has to be considered. The study is carried out to realize the heterogeneous database sharing for educational resources using multi-media educational resources as the researching object. XML is applied as middleware for the practical requirements of education. The study has important practical significance for the intellectualization of educational and teaching resource platform.

***Keywords*:** *Heterogeneous Database, XML, Education Resource*

1. Introduction

Nowadays, material resource, energy resource and information resource are three pillar resources during the development of technology and economics. Educational information resource, as an important part of information resource, plays an important role in improving the teaching quality and mining the potential of education. Therefore, many countries have set up national educational resources centers, such as the National Educational Resources Information Center of United States, the National Network for Basic Education Resource held by Basic Education Curriculum Development Center of Ministry of Education and Central Audio-Visual Education Center of China. The construction of educational resources base with corporations such as Resource of China School, K12, Clever, ZhongJiaoYuxing, Cisco Tong Fang, Tuteng, Tinghua Tangfang, becomes more and more mature. Education resource has already become an important part of network resource. It plays important role in solving the problems such as information selection, information identification, information digestion, and information individuation when users get amount of information through education resource base.

It turns out to be the most potential resource for global information transferring and sharing with the rapid development of WWW. The requirements and development of new fields, such as e-commerce, e-books and distance education have made Web data more complex and diverse. Therefore, it is difficult to store and manage all the different Web data by traditional database technology. XML is becoming the data description and exchange standard on Internet. Meanwhile, XML and a series of related standards have been widely accepted and used, including the generation, storage, analysis of XML documents, which lay the foundation for XML as database, as well as offer the possibility to realize the sharing of data in heterogeneous databases. The automatic transforming XML into different relational databases effectively has different ways. Oracle XML SQL Utility models XML documents element as a group of nesting tables, through the element that oracle object data type modeled; IBM DB2 XML Extender saves the XML documents as BLOB type object, decomposes them to a group of tables, and defines the XML collection by the XML1.0 grammar. Microsoft solves the problem through expanding SQL-92 and involving the OPENXML line collection; Sybase Adaptive Server takes ResultSetXml Java class as the foundation of processing XML documents in two directions [1–3].

However, all manufacturers have a general character that the XML durability establishes in a special foundation, and there is no general facility to save all the XML documents. If the XML documents use a new grammar, it needs a special mapping. This is very disadvantageous for the visiting and sharing the different heterogeneous education resource. Firstly, education resource's description is very complex, each education resource's comprehensive description reaches more than 160 terms based on international standard, which has the internationalization request, and needs a standardized description frame. Secondly, the existing education resource's description is not normal, on the one hand we need to preserve its original description; on the other hand, we need a stan-

dardized description mapping. Thirdly, when users use an education resource, its description must personalize to meet the users' especial requirements by cutting out and transforming its description. Fourthly, education resource's description attribute value is dissimilar from different understanding aspects based on education resource belonging to different disciplines. Fifthly, the education resource has very wide manifestation, including text, image, sound, video, animation and so on. Its respective description must reflect both the general character and the difference. Only using existing tools given by manufacturers can not solve these problems. This article takes the XML documents as the middle data exchanging model to study the sharing problem of education resource heterogeneous database by taking the multimedia education resource as the study object, embarked from the actual requirements of education teaching.

2. Sharing Technology Selecting for Heterogeneous Database of Education Resource

Heterogeneous database systems [4] are collections of many related database systems which could achieve sharing and transparent accessing the data. Each database system with its own DBMS has already existed before adding in the heterogeneous database system. All components of the heterogeneous database have their own autonomy; each database system still remains its own application characteristics, the integrity control and security control when sharing the data. After carrying out data conversion, on the one hand, all the information to be shared is converted from source database to the destination database; on the other hand, such a conversion can not contain related redundant information. We use the interoperability technology and data integrity technology of heterogeneous database to achieve the tasks of data sharing in heterogeneous database. Heterogeneous databases interoperability is a prerequisite for data integration. The core is data accessing, so as to provide underlying technology for achieving data integration [5–6].

The methods used to achieve the exchanging of data among databases are various. Development tools with data transfer tool, such as data pipeline in the PowerBuilder, can be used, but it relies on the database structure, and its flexibility is poor. E-mail system can not meet the data exchange requirement between heterogeneous database systems, but you must connect two databases before data transmission, which in turn lower its flexibility. We solve the referred problems by using XML as a middleware of heterogeneous databases system. First of all, as a middleware, XML makes heterogeneous database system independent, and incompatibility of heterogeneous database systems is solved. If only front desk application can support XML, it can transfer the exchange of information among heterogeneous database systems to mapping relationship between XML. Secondly, XML documents are easy to read and modify. XML documents could be opened and modified by an ordinary WordPad, and the structure is relatively simple which is easy to read, modify and convert. All these increase the flexibility of information exchanging heterogeneous database system and the scalability of XML. Thirdly, XML document format is simple, which reduces the complexity in the process of programming procedure, lowers the workload of programmers, and XML documents' code is easy as well. In view of this, we choose XML as a middleware to achieve data exchange among heterogeneous database systems [7–9].

In this research, education resource in the database includes image, text, video, music, flash and so on. Database's heterogeneities are mainly the DBMS heterogeneities including Oracle database, IBM DB2, Sybase database, Microsoft SQLserver2000 and MySQL. These already operated database systems display differently in many aspects, including data type definition, data access mode, data manifestation and so on. Therefore different database management systems cannot be connected directly to exchange the information. Oracle database, IBM DB2, Sybase database, Microsoft SQLserver2000 and MySQL have provided powerful support to XML. We transform the communication between various business databases into the data transformation between various business databases and the XML, and take image, video, text, music and flash deposited in various business databases into XML documents and import all the data into the dynamic standard database. Therefore, we adopt the following technical option. We preserve the normal primitive tree data by using various manufacturers mapping rule. The data processing frame is established. The database clusters are controlled by code distributing in the pure application procedure, business process, the database level and the application logic between the storage process, and the data processing frame guiding layer is formed by taking the education resource international standard as the foundation. The dynamic unification education resource database is established based on education resource's multiplicity and the education resource description's multiple perspectives of different discipline. User's personalized request is obtained by cutting out unification education resource database with data processing frame.

3. Education Resource Database System Heterogeneous structure

The design of the system structure is divided into three points: the presentation layer, the database layer and the logic layer, as shown in Figure 1.

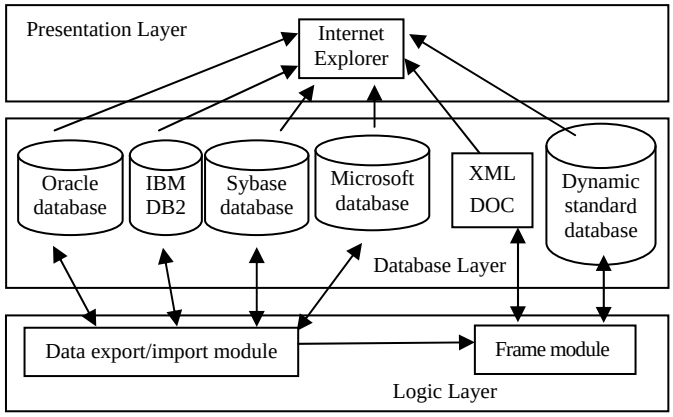


Figure 1. The design of system structure

imaID	Type	Major	Object	Source	Author	TTime	Key Word	Formats	Lenght	resolution	Sweep	Color
01001	animal	Bio-engineering	Graduate Students	Southeast University	Na Tang	2007/05	yeast, bacteria	gif	36	640*480	72	002244
01002	figures	history	Undergraduate Students	Nanjing University	Wei ZHAO	2004/06	background, achievements	jpeg	80	1024*768	300	FF0022
01003	equipment	sports	Undergraduate Students	Suzhou University	Liang-Qian Li	2002/06	function, using methods	jpeg	45	320*240	72	000044
01004	board	computer	Graduate Students	Yancheng Teachers University	Xia Wang	2007/06	CPU, graphics, memory	jpeg	68	1024*768	350	110033
01005	monitor	computer	Undergraduate Students	Wuhan University	Xiaoli Wang	2001/06	LCD, brightness	jpeg	55	1024*768	300	2244CC
01006	natural	geography	Graduate Students	Southeast University	Yao Wu	2000/06	Earthquake, plate movement	gif	38	640*480	72	320C14
01007	plant	medicine	Graduate Students	Southeast University	Hang SUN	2005/02	panda, artificial rearing	gif	74	1280*1024	300	05232B
01008	animal	computer	Undergraduate Students	Peking University	Xia Li	2005/06	lily, photosynthesis	jpeg	38	750*453	72	204A15

Figure 2. Education resource data

The presentation is the browser. Users view the results and the displaying form through browser.

The database layer is composed by Oracle database, IBM DB2, Sybase database, Microsoft SQLserver2000, MySQL, dynamic standard database and the XML documents. We attribute XML document to this layer although XML itself is not a database, since it can be seen as a complete database system with some other tools.

Logic layer consists of two modules: data export/import module and the education resource data frame module. Data export module's function is to export various business databases data and produce corresponding XML documents. Data import module's function is to analyze XML documents and import the data to various business databases. The education resource data frame modular controls all database clusters, and it is responsible to establish the dynamic unification education resource database.

4. Implement of Heterogeneous Database of Education Resource

Oracle XSU draws the XML documents to DOM, and decomposes the XML documents into a group of sub-documents by using XSLT. IBM DB2 XML Extender establishes the mapping through DAD between the database table and the XML documents' structure, storing by DB2 CLOB data type. Microsoft SQLserver OPENXML uses sp_xml_preparedocument as storing process, gaining a XML documents handle through translating XML documents into the internal DOM expression. Sybase uses XML documents type ResultSet to describe a XML documents metadata and the actual line data. The selected data is shown in Figure 2.

Document Object Model is a set of standards set by W3C, which provides an interface parsing the document. Various program languages achieve these interfaces in accordance with the DOM standards, and the parser is


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
-<image>
   -<imageinformation>
      <imaID.> 01001</imaID>
      <Type> animal</Type>
      <Major> Bio-engineering</Major>
      <Object> Graduate Students</Object>
      <Source> Southeast University</Source>
      <Author> Na Tang</Author>
      <TTime>2007/05</TTime>
      <Key Word>yeast,bacteria</Key Word>
      <Formats> gif</Formats>
      <Lenght> 36</Lenght>
      <Resolution>640*480</Resolution>
      <Sweep> 72</Sweep>
      <Color> 002244</Color>
     <Address>..\\DataSharing\src\image\1.gif</ Address >
      </imageinformation>
-<image>
   -<imageinformation>
      <imaID.> 01002</imaID>
      <Type> figures</Type>
      <Major> history</Major>
      <Object> Undergraduate Students</Object>
      <Source> Nanjing  University</Source>
      <Author>Wei Zhao</Author>
      <TTime> 2004/06</TTime>
      <Key Word> background, achievements</Key Word>
      <Formats> jpeg</Formats>
      <Lenght> 80</Lenght>
      <Resolution> 1024*768</Resolution>
      <Sweep> 300</Sweep>
      <Color> FF0022</Color>
     <Address> ..\\DataSharing\src\image\2.gpeg</ Address >
      </imageinformation>
```

Figure 3. Education resource spanning tree

given to parse the documents. The parser establishes a tree in memory through reading XML documents. The tags of XML document, tagged text content and entities correspond to a certain node of the tree in the memory. It's easy to deal with XML documents, to read, traverse, modify, add and delete the documents through operating the node of tree in memory. We use DOM parser to do XML programming for an application, which can easily handle XML documents by operating the node of tree in memory to obtain the data needed.

DocumentBuilderFactory class is responsible for creating the instance. DocumentBuilderFactory class calls its newInstance () method to instantiate a DocumentBuilderFactory object. Factory object calls newDocumentBuilder () method to return a DocumentBuilder object. And finally the builder object calls newDocument () method to achieve instantiating Document interface.

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder =
factory.newDocumentBuilder();
Document doc = builder.newDocument();
doc.setXmlVersion("1.0");
Element root = doc.createElement("image");
doc.appendChild(root);
```

The CreateXML.xml documents are produced as follows:

```
File file = new
File("F:/temp/MatPrj/WebRoot/CreatXml.xml");
if( !file.exists()||!file.isFile()){
new FileOutput-
Stream("F:/temp/MatPrj/WebRoot/CreatXml.xml");
file = new
File("F:/temp/MatPrj/WebRoot/CreatXml.xml");
}

StreamResult streamResult = new StreamResult(file);
Source inputSource = new DOMSource(doc);
TransformerFactory transformerFactory =
TransformerFactory.newInstance();
Transformer transformer =
transformerFactory.newTransformer(); trans-
former.transform(inputSource, streamResult);
```

The XML documents are produced, and the data is imported to XML documents, its education resource spanning tree is shown in Figure 3.

The data processing frame is established to control the database clusters, and the dynamic uniform education

WU.dbo.image table

imaID	Type	Major	Object	Source	Author	TTime	Key Word	Formats	Lenght	resolution	Sweep	Color
01001	animal	Bio-engineering	Graduate Students	Southeast University	Na Tang	2007/05	yeast, bacteria	gif	36	640*480	72	002244
01002	figures	history	Undergraduate Students	Nanjing University	Wei ZHAO	2004/06	background, achievements	jpeg	80	1024*768	300	FF0022
01003	equipment	sports	Undergraduate Students	Suzhou University	Liang-Qian Li	2002/06	function, using methods	jpeg	45	320*240	72	000044

WU.dbo.text table

TxtID	TxtType	TxtMajor	TxtObject	TxtSource	TxtAuthor	Txt Time	TxtKey Word
02001	Economics Paper	International economic	Graduate Students	Economics	Li Zhang	2007/05	Bubble economy, Economic globalization
02002	Economics Paper	China's economy	Undergraduate Students	Economics	Ke Ban	2004/06	China imported inflation, Real estate and living
02003	Management Paper	Business Management	Undergraduate Students	Management	Xiao-ming WU	2002/06	Enterprise Project Management

WU.dbo.vedio table

VedID	VedType	VedMajor	VedObject	VedSource	VedAuthor	VedTime	VedKey Word
05001	Computer	An Introduction to Computer	Graduate Students	Southern Yangtze University	Kequn Wang	2006/05	computer, chip
05002	Computer	Computer Application	Graduate Students	Suzhou University	Liang Ke	2004/06	computer, NET,J2EE

WU.dbo.flash table

FlaID	FlaType	FlaMajor	FlaObject	FlaSource	FlaAuthor	FlaTime	FlaKey Word
03001	Computer	Software Engineering	Graduate Students	Shenyang Polytechnic University	Xing Liu	2005/05	computer, C++
03002	Education	Chinese Language & Literature	Underraduate Students	Central China Normal University	Chuan Zhao	2004/06	literature, education

WU.dbo.music table

MusID	MusType	MusMajor	MusObject	MusSource	MusAuthor	MusTime	MusKey Word
04001	Education	Chinese Language and Literature	Graduate Students	Southeast University	Keyi Zhang	2007/05	literature, education
04002	Education	Chinese Language and Literature	Undergraduate Students	Nankai University	Guosheng Huang	2003/06	literature, education

Figure 4. Dynamic uniform education resource database

resource database is also established as shown in Figure 4 according to the multiplicity of education resource's manifestation and the multiple perspective of the education resource description of different discipline based on international standard of education resource.

The personalization description of education resource is obtained by cutting the dynamic unification education resource database using the data processing frame according to the user's personalized request, as shown in Figure 5.

Imageinformation 0
imaID 01001
Type animal
Major Bio-engineering
Object Graduate Students
Source Southeast University
Author Na Tang
Ttime 2007/05
Key Word yeast, bacteria
Formats gif
Length 36
Resolution 640*480
Sweep 72
Color 002244
Address ..\\DataSharing\src\image\1.gif

Figure 5. The personalization education resource result in client after cutting out

5. Conclusions

The paper defines a XML document which describes the database structure. We fill the information of the education resource database into self-explanatory XML documents in order to let users create database friendly. And a scheme is made as designing a middleware between the database and its outside. A data processing frame is advanced to process each kind of actual problems which are brought by the complexity, the dynamic and the personalization of the education resource data. The exchanging requests of internal or external education resource data are submitted to the middleware with XML. The interaction of specific education resource database is realized by the middleware and the results are fed back to the requester through XML. All the details in the process have been shielded to achieve transparent sharing access of heterogeneous education resource database.

REFERENCES

[1] Oracle XML-SQL Utility, http: // www. oracle.com /technology/ index.html. Last accessed on July 27, 2009.

[2] IBM DB2 XML Extender, www.ibm.com/ software/ data/ db2/ extenders/xmlext. Last accessed on July 27, 2009.

[3] XML Perspective, In control with FOR XML Explicit. SQL Server Magazine, http://msdn.mcrosoft. com/ library/ periodic/. Last accessed on July 27, 2009.

[4] A. P. Sheth and J. A. Larson, "Federated database systems or managing distributed, heterogeneous, and autonomous databases," ACM Computing Survey, Vol. 22, No. 3, pp. 183–236, 1990.

[5] Q. Feng, H. Q. Lv and H. Feng, "The connection of Heterogeneous databases," Computer and Information Technology, Vol. 9, 2001.

[6] X. Wang and S. M. Wei, "Java technology-based distributed heterogeneous database Web access technology," Computer Engineering and Applications, pp. 135–138, 2000.

[7] D. Martin, "XML High-level Programming," Beijing Machinery Industry Press, pp. 77–103, 2001.

[8] D. Motton, "XML programming technology [M]," Beijing Machinery Industry Press, pp. 134–200, 2001.

[9] X. H. Dong [EB/OL], http://www.XML.org.cn, Applications to build XML.

J. Software Engineering & Applications, **2009, 2: 206-208**
doi:10.4236/jsea.2009.23028 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

Scientific Research

MicrobIdentifier: A Microbial Identification Software Based on Mass-Spectrometry

Feng LIU, Lu LI, Chi ZHANG, Lingbing WANG, Pei LI

International School of Software, Wuhan University, Wuhan, China.
Email: wolflf@126.com, {lulu.li1989, chzhcn88}@gmail.com

Received May 18th, 2009; revised July 5th, 2009; accepted July 16th, 2009.

ABSTRACT

As the technology of microbial identification by mass cataloging has been widely used, we have developed the microbial identification software, MicrobIdentifier, which integrates and automates different steps in the procedure of rapid species identification based on mass-spectrometry. This software is written in Java for cross-platform intention.

Keywords*: *Microbial Identification, Mass-Spectrometry

1. Introduction

With the development of the technology, microbial identification by mass cataloging has attracted considerable attention due to its high efficiency and automation. In order to improve efficiency and automation of this technology, we've developed this microbial identification software based on the spectral coincidence function proposed in [1]. The software has two major functions: First, it can be used to search for all the possible primer pairs among the given genes of different species, and evaluate these primer candidates by giving each pair a score. This is proved to be a useful reference during primer design. Second, it takes advantage of the spectral coincidence function to compare mass spectrometric observables with theoretical fragmentation patterns, and further to determine the genetic affinity between the sample gene and genes of known species in the database. This will free researchers from the effort of comparing the fragmentation patterns manually.

2. Algorithm

The core algorithm our work has been based on is a spectral coincidence function proposed in [1] as follow:

$$C_{ij} = C\left(M_i, M_j\right) = \frac{2 \times M_i \cdot M_j}{(M_i \cdot M_i) + (M_j \cdot M_j)}$$

The dot-product in the coincidence function is defined as

$$\langle M, M' \rangle = M \cdot M' \equiv \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \delta\left(m_i - m'_j\right)$$

where M is the mass vector of one sample's fragmentation, which has N_1 elements with m_i standing for the ith element, while M' is the mass vector of the other sample, which has N_2 elements with m'_j standing for the ith element. The discrete delta function δ is:

$$\delta(k) = \begin{cases} 1 & k = 0 \\ 0 & otherwise \end{cases}$$

Based on the formulas, the inner-product is greater if the two samples have more fragmentation of the same mass. The coincidence function normalizes the inner-product value to a range between zero and one, and a high value of the coincidence function indicates more similarity between the two genes in comparison. Therefore, this function can be used to score the similarity in both the primer search process and the identification process.

The algorithm in primer search process is as follow:

1) Align all the gene sequences with ClustalW algorithm [3].

2) Find regions where all the sequences have more than N nucleotides at the same place and in the same order, which are the conserved regions. If the regions are less than two, then exit.

3) Take two conserved regions and check whether the number of nucleotides is more than M. Take another pair of regions if otherwise.

4) Cut the regions between two conserved regions (conserved regions included) after every "G", filtering the fragments which have less than L nucleotides.

5) Calculate the mass of all fragments of each sequence, and then form the sequence's mass vector.

6) Take the mass vectors of one pair of gene sequences and calculate the score indicating their similarity by using the coincidence function.

7) Repeat Step 6 until any pair of all the gene sequences has been compared. Calculate the average value of all the scores calculated in Step 6. The average value is the final score of the primer pair chosen in Step 3.

8) Repeat the steps from 3 to 7 until all the combinations of the conserved regions are considered.

Optimal primer pairs are those conserved regions with very variable regions in between. A primer pair with a lower score is better than the ones with higher scores, since there is less similarity between the primer pairs, thus the test samples could be identified with much more ease in the identification process.

The algorithm in identification process is almost the same as the Steps from 3 to 6 in the primer search process with one exception that, in identification process, it is the comparison of experimental data and the computed mass vector in the database. A higher score indicates more genetic affinity, suggesting a higher possibility of being the same species.

Given inevitable experimental inaccuracy, the discrete delta function δ is further modified to be:

$$\delta(k) = \begin{cases} 1 & |k| < tolerance \\ 0 & otherwise \end{cases}$$

Thus, tolerable difference between masses is ignored.

3. Software

The software accepts a fasta file as input, then invoke a new process running clustalw that also takes the .fasta file. As long as the .fasta file is valid in format, a .aln file, the result of clustalw's pairwise alignment, is created and afterwards captured. Through parsing both the fasta file and .aln file, a data group is fabricated. In the software, a data group is a concept of a pool of sequences with user configuration that is identification-ready. Typically users need to assign four thresholds: the minimum length of a sequence fragment after simulated cutting; the minimum length of a primer; the minimum and maximum length of the variable region between primer pairs. The same sequence pools with different configurations are different data groups. The software ensures users only work on one data group at a time given that the concept of data group supports sufficiently in flexibility and reusability for users to handle microbial identification merely on one data group in most situations. During this preprocessing phase, the software stores user configurations as well as the data group sequences into the database for the purpose of 1) enabling access to previously processed data groups in later cases 2) providing thresholds reference for identification process.

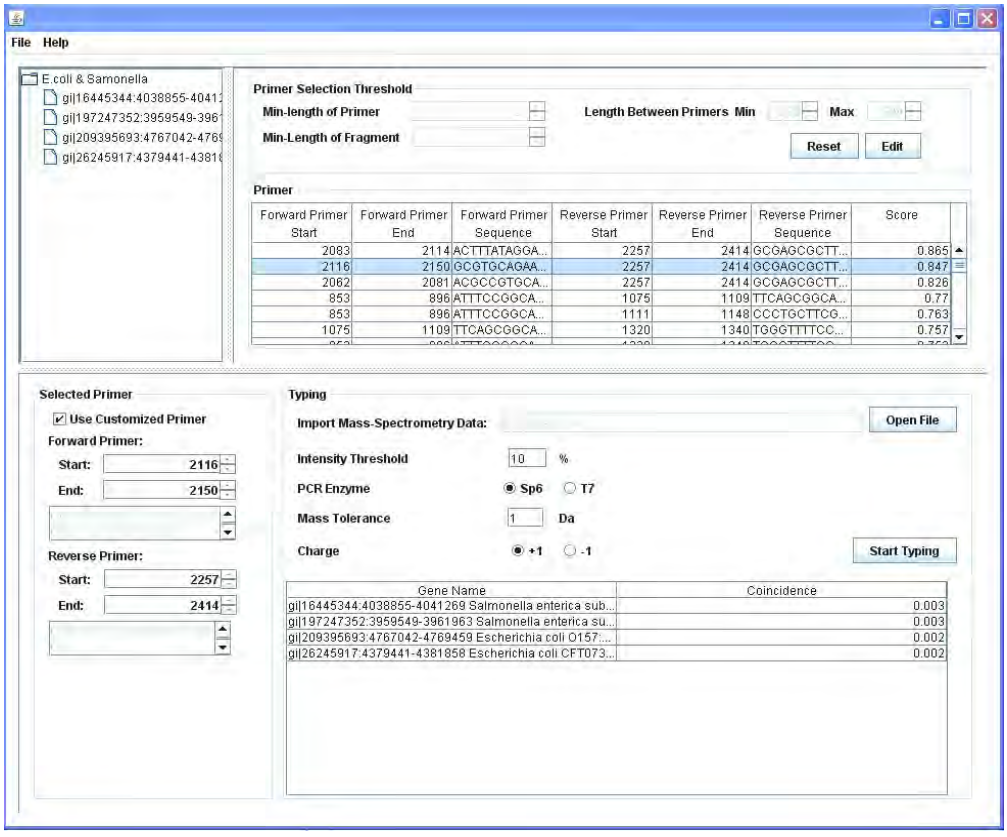


Figure 1. MicrobIdentifier screenshot

The user interface shows the sequences in the pool; primer selection thresholds and primer pair candidates are also given out if current data group is loaded from database, whose primer pair candidates have already been worked out after proper configuration in previous use. The more usual case, however, is the user sets up basic configuration after a new pool is given, parsed down and shown on UI, to calculate potential primers pairs. The list of primer pairs is sorted by score in ascending order. The configurations are saved into the database in associate with the working data group.

To perform microbial identification, the software uses exported ASCII Spectrometry .txt file from DataExplorer, whose data is the mass spectrometry result from MALDI-TOF. Users are free to customize proposed primer pair candidates to choose a subset, however mandatory to provide some parameters about the conditions in their mass-spectrometry experiment, including: in vitro transcription enzyme, either SP6 or T7; mass tolerance and minimum intensity threshold; whether the electric charge is positive of negative during MALDI-TOF experiment. The software parses the input file, generates peek list after filtering peak values below the intensity threshold, taking into account the experimental inaccuracy by means of adopting tolerance and finally provides the identification consequence.

Figure 1 shows the interface of MicrobIdentifier.

4. Acknowledgements

This paper is sponsored by the National Science and Technology Major Project 2009ZX10004-107 and The Natural Science Founds of Wuhan University F020504.

REFERENCES

[1] G. W. Jackson, R. J. McNichols, G. E. Fox and R. C. Willson, “Bacterial genotyping by 16S rRNA mass cataloging”, BMC Bioinformatics, vol.7, pp. 321–335, June 2006.

[2] Z. D. Zhang, G. W. Jackson, G. E. Fox, and R. C. Willson, “Microbial identification by mass cataloging,” BMC Bioinformatics, Vol. 7, pp. 117–135, Match 2006.

[3] J. D. Thompson, D. G. Higgins, and T. J. Gibson, “CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice,” Nucleic Acids Research, Vol. 22, pp. 4673–4680, September 1994.

[4] C. Honisch, Y. Chen, C. Mortimer, C. Arnold, O. Schmidt, D. van den Boom, C. R. Cantor, H. N. Shah, and S. E. Gharbia, “Automated comparative sequence analysis by base-specific cleavage and mass spectrometry for nucleic acid-based microbial typing,” Proceedings of the National Academy of Sciences, Vol. 104, pp. 10649–10654, June 2007.

[5] H. Steen and M. Mann, “The abc’s (and xyz’s) of Peptide Sequencing,” Molecular Cell Biology, Vol. 5, pp. 699–711, September 2004.

J. Software Engineering & Applications, 2009, 2: 209-220
doi:10.4236/jsea.2009.23029 Published Online October 2009 (http://www.SciRP.org/journal/jsea)

An Exploratory Case Study in Designing and Implementing Tight Versus Loose Frameworks

Manjari GUPTA[1], Ratneshwer GUPTA[2], A. K. TRIPATHI[3]

[1]Department of Computer Science, Faculty of Science, Banaras Hindu University, Varanasi, India; [2]Department of Computer Science, MMV, Banaras Hindu University, Varanasi, India;[3]Department of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India.
Email: {manjari, ratnesh, anilkt}@bhu.ac.in

Received May 5th, 2009; revised June 20th, 2009; accepted June 24th, 2009.

ABSTRACT

Frameworks provide large scale reuse by providing skeleton structure of similar applications. But the generality, that a framework may have, makes it fairly complex, hard to understand and thus to reuse. Frameworks have been classified according to many criteria. This paper proposes two types of framework (based on the concept of 'generality') named as: tight framework and loose framework. A case study is done by developing loose and tight frameworks for the application sets of Environment for Unit testing (EUT) domain. Based on the experience that we got by during this case study, we tried to find out the benefits of one (tight or loose) framework over the other. This work attempts to provide an initial background for meaningful studies related to the concept of 'Design and Development of Framework'.

Keywords*: *Framework Reuse, Environment for Unit Testing, Condition Coverage Criteria

1. Introduction

"Frameworks are reusable designs of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate". It is always the result of domain analysis [1]. Frameworks may be classified according to many criteria such as manner of deployment in different applications, level of support that provides to applications, type of services they have (entity framework, control framework), level of abstraction they have (white box framework, black box frameworks) etc. Frameworks are normally developed by keeping in mind requirements of multiple similar applications. Frameworks should be developed and delivered in such a manner so that problems in instantiation (like determining the applicability of a framework, understanding and modifying if necessary, architectural mismatch etc. [2]) do not arise and overheads (requirements that are not required in a particular application), do not get transported with the framework. In order to develop a framework, its scope must not be an afterthought and it should be considered at the beginning i. e. at the time of designing frameworks. When one talks about frameworks, its scope and generality are necessary to consider.

Here, we classify frameworks by considering the generality they have. In software engineering literature, we could not find the formal categorization of frameworks based on 'generality' concept. We classify frameworks in two categories by considering their generality as follows.

A ***loose framework*** is a framework that does not fix the way of performing many activities (that may be performed differently in similar applications in a domain) in the framework itself. It only provides the control abstraction and thus may need to work together with other frameworks for some activities that will extensively interact with it. Such frameworks may be useful for those applications that have many possible variations in their requirements for example business application systems, E-governance systems etc.

A ***tight framework*** fixes the way of performing most of such activities in the framework itself. Thus, these frameworks are highly useful for (only) those similar applications that require performing those activities in a particular way as defined and implemented in the framework. Such frameworks may be useful for those applications that have very few variations in their properties like embedded systems, pervasive systems etc.

During design of a framework certain roles and responsibilities amongst the classes along with their collaboration are fixed. Variability among applications is represented as hot spots. Thus, by comparing the already fixed roles and responsibilities as well as the variability (hotspots) of different frameworks, for the same domain, one can say "what is the scope of a framework?" and

thus "whether a tight or a loose framework development would be beneficial for that domain?"

To the best of our knowledge, there is lack of such study/work in which tight and loose frameworks, for a same domain, are compared so that one can list the cases in which one (tight or loose framework) would be better than the other. In this paper, a tight and a loose framework for 'Environment of Unit Testing' have been developed and a comparative study has been made between the two types of frameworks. Based on the observations of this study, we tried to answer the following questions:

1) Which framework (tight or loose) is more reusable in terms of "ease of reuse"?

2) Which one is more reusable in terms of "number of reuses"?

3) Which one is easy to develop?

4) Which one is heavier in terms of size?

5) Which one is more complex?

Reminder of this paper is organized as follows. The section 2 attempts to present, in a concise manner, the research efforts related to the topic of discussion. In section 3, we briefly describe the domain 'EUT' for which frameworks have been developed. Designs of a loose and a tight framework for "EUT" have been described in section 4 and 5 respectively. A comparative study of loose and tight frameworks has been discussed in section 6. Finally, we conclude in section 7.

2. Related Work

Software engineering, over the last decades, has been promoting the development of software systems with software frameworks. Researchers and practitioners have been considering various aspects of framework development and related issues. Software frameworks are classified according to several criteria.

There are two styles of frameworks that are commonly used: called and calling frameworks. Sparks et al. [3] showed the reuse with Called and calling frameworks. *Called* frameworks are very much like traditional libraries in that the application code calls the framework when some framework service is needed. *Calling* frameworks on the other hand, reverse the role of the framework and the application, because the framework calls the application code, rather than the other way around. Some authors defined frameworks according to domain dependency: *vertical* and *horizontal* frameworks [4]. A framework dependent on specific domain is referred to as *vertical framework*. A framework independent on specific domain is referred to as *horizontal framework*. According to Taligent, Inc (now IBM) [5]) the problem domain that a framework addresses can encompass application functions, domain functions, or support functions. Application frameworks encapsulate expertise applicable to a wide variety of programs. These frameworks encompass a horizontal slice of functionality that can be applied across client domains. Current commercial graphical user interface (GUI) application framework, which supports the standard function required by all GUI applications, is one type of application frameworks. Domain frameworks encapsulate expertise in a particular problem domain. These frameworks encompass a vertical slice of functionality for a particular client domain. Examples of domain frameworks include: a control systems framework for developing control applications for manufacturing, securities trading framework, multimedia framework, or data access framework. Support frameworks provide system-level services, such as file access, distributed computing support, or device drivers. Several authors defined Frameworks based on the techniques used to extend them: White-box, Black-box and gray-box frameworks [4]. In a white box framework, the framework user is supposed to customize the framework behaviour through sub-classing of framework classes. On the other hand, a black box framework user does not have access to framework code. Gray-box frameworks lie between white and black box framework. Frameworks are also classified by their scopes: *System infrastructure frameworks*, *Middleware integration frameworks* and *Enterprise application frameworks***.** *System infrastructure frameworks* simplify the development of portable and efficient system infrastructure. Communication frameworks, proposed by Schmidt [6], also belong to *System infrastructure frameworks. Middleware integration frameworks* are commonly used to integrate distributed applications and components. Common examples include ORB frameworks, message-oriented middleware, and transactional databases. *Enterprise application frameworks*, proposed by Fayad et al. [7], address broad application domains such as telecommunications, avionics, manufacturing, and financial engineering.

However, no such study has been done till now to develop different types of frameworks for a domain and compare them to show in which situation which one is more applicable for reuse. We had earlier done risk analysis in framework development and its reuse [8]. This paper attempts to extend the above contributions further by considering the comparative differences of loose and tight frameworks.

3. Environment for Unit Testing

We briefly explain the domain "EUT" for which frameworks are developed in the next sections. To develop a framework, it is necessary to understand the domain for which it is to be developed. Thus, first we briefly describe the unit testing process.

Unit testing is a dynamic method for verification, where the smallest unit of software design- *the software component or module* (unit code) is actually compiled and executed. The unit testing focuses on the internal processing logic and data structures within the boundary

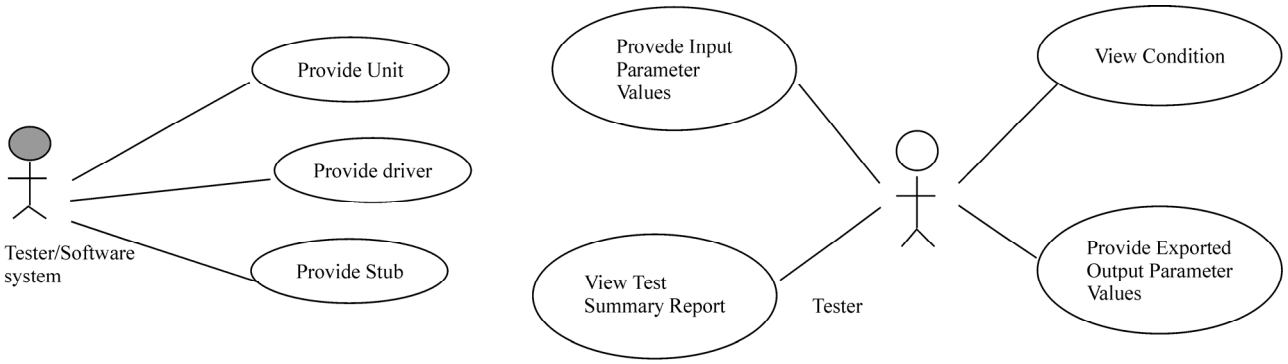


Figure 1. Use case diagram for tight framework

of a component [9]. As the focus of this testing level is on testing the code, *structure testing* is best suited for it. Unit testing commences with generating test drivers and stubs for the unit. Next, test cases are generated. A test case is a set of test inputs, on which the unit code, to be tested, is executed. The output of the program for each test case is evaluated to verify compliance with the corresponding requirement using test oracles. During unit testing several test deliverables (test case specification, error report and test log etc.) are generated. At last, test summary report is generated that specify the result of testing process.

To test the structure of a program, structure testing aims to achieve test cases that will force the desired coverage of different structures. Various criteria have been proposed for this. Most common *structure based criteria* are based on the *control flow* of the program for example statement coverage, branch coverage, decision/condition coverage and path coverage.

We developed both of these (tight and loose frameworks) for the domain of 'EUT'. Both of these frameworks test a unit written in C language. In the tight framework we fixed the test case generation activity (based on condition coverage criteria). Because of that, this framework can only be used if the unit testing criteria is 'condition coverage'. For rest of the unit testing criteria, this reusable framework is useless. However, the test case generation activity was not fixed in the loose one and thus any test case generator (developed by considering any unit testing criteria) that can generate test cases to test a C unit can be integrated with the loose framework. Both the frameworks (tight and loose) have been developed in C++ language.

4. The Proposed Tight Framework for 'EUT'

We first describe the design and implementation of tight framework for 'EUT'.

The tight framework is developed by considering the test cases generation based on the *condition coverage* criterion. Thus, this framework supports the development of a family of applications that would test a unit based on the condition coverage criteria but differ in the way of getting unit, driver, stubs. For example, a system developed by using this framework may accept these from a human being while others may get these from software systems (that will be generating these automatically).

As we know, any object oriented software and in particular any object oriented framework is a collaboration of domain, control logic, utility and interface classes. In this remaining section and in the next section, we identify these classes for both tight and loose framework for 'EUT' and show how they collaborate with each other.

4.1 Analyzing the Requirements of Tight Framework for 'EUT'

By *studying the above problem statement*, we specify how a user will interact with this tight framework in the *use case diagram* shown in Figure 1. Gray ovals show the functionalities that are needed to be customized and black ovals show functionalities that are prefixed in the framework. Way to provide unit to be tested, drivers and stubs (if any) to the framework (manually by the tester or automatically using a software system) may be different for different applications that will be developed by using this framework. Thus, in the use case shown on the left hand side of Figure 1, actor may either be a human being or a software system and that's why has shown as gray (as per the convention usually used to describe a framework). As shown in the use case on the right hand side of Figure 1, each condition, to be tested, is displayed to the tester and then a tester provides test cases corresponding to each condition. These test cases are run and test summary report is delivered to the tester after the completion of the testing process.

From the problem specification, described above, we can identify the following classes:

Unit – is an important class that would be tested using this framework,

Driver – would invoke and provide environment for the execution of the unit (if required),

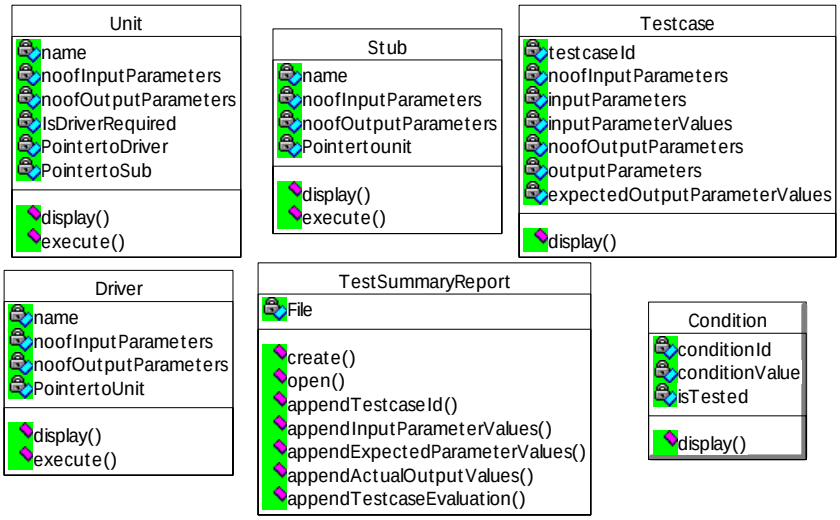


Figure 2. Domain classes for tight framework

Stub – would be called by the unit during its execution (if required),

Test case – is a class that would represent a test case,

Condition – is a class that represents a condition to be tested and

Test summary report – represents the test summary report delivered to the tester after testing. These domain classes along with their attributes and methods are shown in Figure 2.

4.2 Designing the Tight Framework for 'EUT'

To represent the abstract dynamic behavior of the system developed as a tight framework, we first describe scenarios.

The success scenario is as follows:

1) Unit is provided (way would be specific to the application) to the system.

2) If driver is needed, it is provided (way would be specific to the application) to the system.

3) If stubs are needed, these are provided (way would be specific to the application) to system.

4) Until all the conditions, in the unit, are tested

a) Next condition is identified.

b) System displays this condition to the tester and asks the number of test cases (N) that need to be generated to test this condition.

c) For this number (N) of times

i) System asks the next test case.

ii) Tester inputs the test cases.

iii) System executes this test case and redirects the output value of the execution to a file.

iv) The actual output value is compared with the expected output value and is shown to the tester.

v) System appends the testing result of this test case (condition, corresponding test case, actual output value, execution status of the test case (successful/unsuccessful, executed or not) etc.) in the test summary report.

vi) System asks whether to proceed further or quit.

vi) If tester wants to quit (in case of getting wrong result to correct the unit), the path and name of the Test Summary Report is displayed to the tester and system stops.

d) System asks whether to proceed further or quit.

e) If tester wants to quit, the path and name of the Test Summary Report is displayed to the tester and system stops.

5) A message "all the conditions have been tested" and the path and name of the Test summary report is displayed to the tester.

An *exception* scenario could be: if the format of any of the information related to the program (unit, driver or stub) is unacceptable by the system, for example if the values of the input parameters, needed as a string containing all these values separated by a space, is not provided in the required format. In all these types of situations, system would display different error messages according to the situation.

Another abnormal scenario could be: if a test case cannot be run successfully and thus the execution fails. System will display an error message to show this situation.

The activity diagrams (Figure 3 (a) and Figure 3 (b)) show the sequence of activities performed during *test case generation* and *execution and comparison of actual and expected output* activities in the framework. All the activities in this tight framework for 'EUT' are shown in Figure 3 (c).

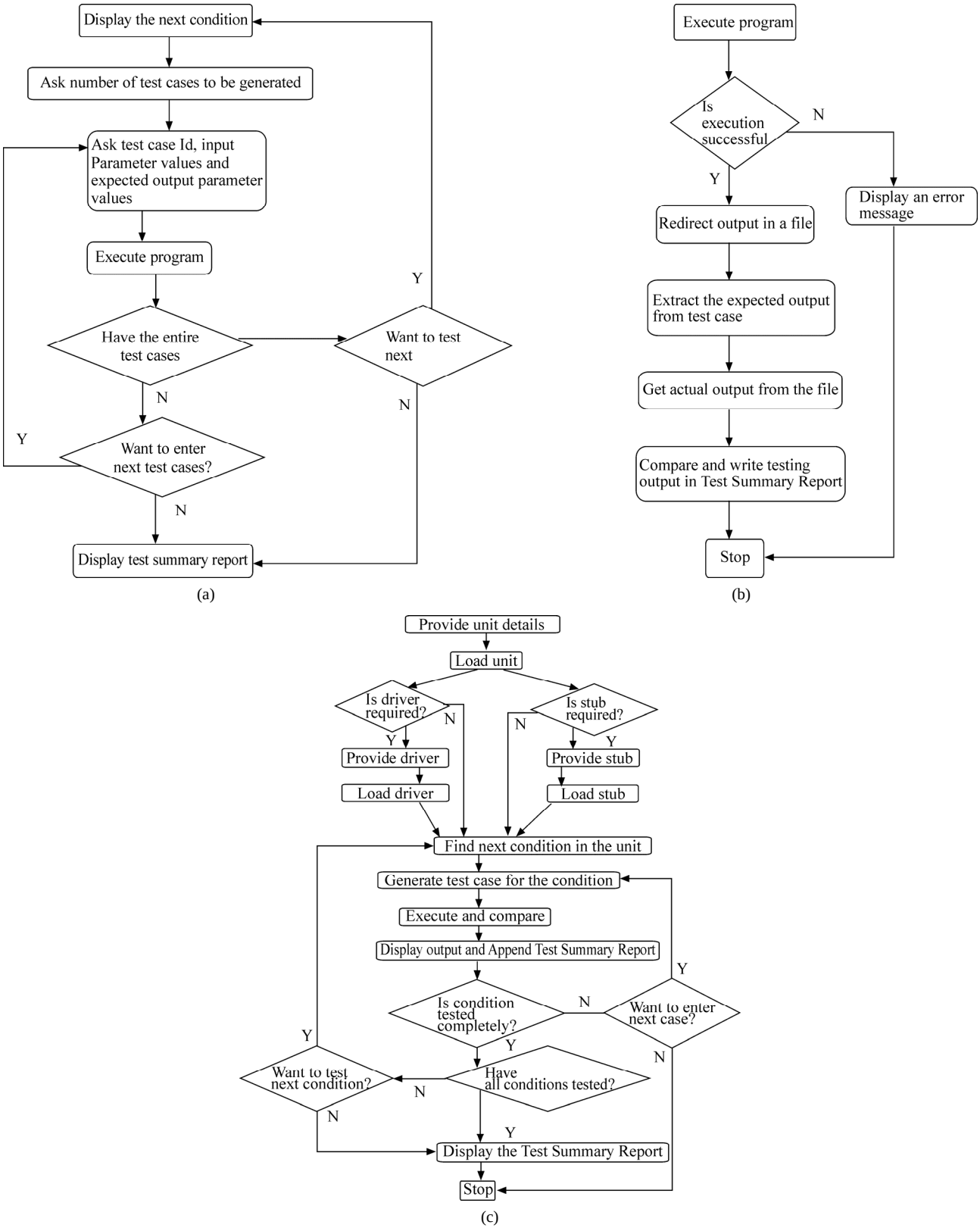


Figure 3. (a): Activity diagram for generating test case for tight framework, (b): Activity diagram for executing and comparing output for tight framework, (c): Activity diagram for tight framework

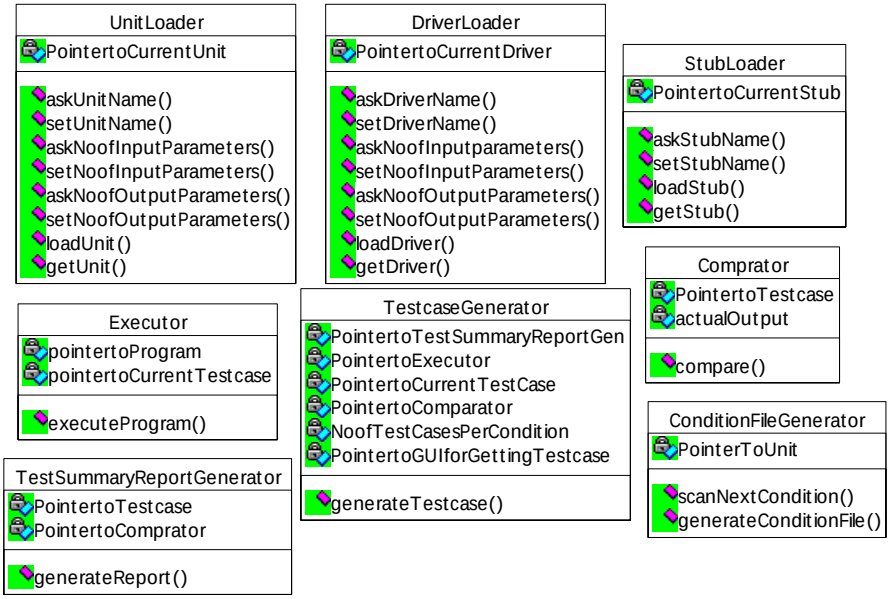


Figure 4. Control logic classes for tight framework

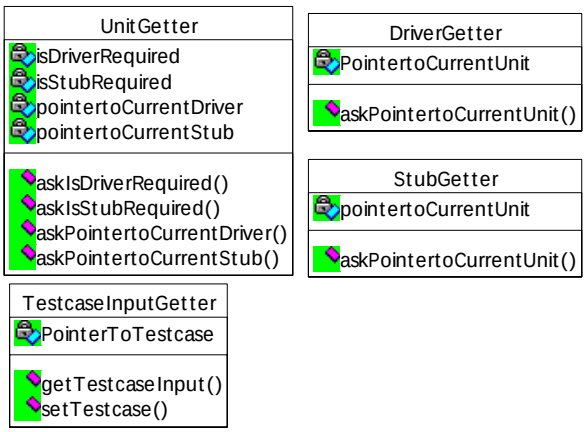


Figure 5. Graphical user interface class for tight framework

Scenarios and activity diagram, described above, hint for the following control *logic classes*:

UnitLoader – responsible for loading the unit to be tested,

DriverLoader – responsible for loading a driver (if any),

StubLoader – responsible for loading a stub (if any),

TestcaseGenerator – responsible for generating test cases with the help of tester,

ConditionFileGenerator – responsible for identifying all the conditions, in the unit, to be tested,

Executor – responsible for executing the unit (if there is no driver) or driver,

Comparator – responsible for comparing the actual output from the expected output parameter values for a given test case and

TestSummaryReportGenerator – responsible for generating test summary report.

These control logic classes along with their attributes and methods are shown in the Figure 4.

Following are the interface classes that are designed for this tight framework. These classes with their attributes and methods are shown in Figure 5.

UnitGetter – responsible for providing unit, to be tested, to the rest of the system.

DriverGetter – responsible for providing driver, if any, to the rest of the system.

StubGetter – responsible for providing stubs, if any, to the rest of the system.

TestcaseInputGetter – responsible for displaying a condition to the tester and getting input parameter values to test that condition.

In this framework Unit, Driver and Stub classes share several attributes and methods thus we take a class **Program** from which all these three classes will inherit properties and operations. Similarly, UnitLoader, Driver Loader and StubLoader also share several properties and operations and we have taken a class **ProgramLoader** that have all the common properties and operations of these classes and these classes are taken as sub class of ProgramLoader class. Further, UnitLoader, DriverLoader, StubLoader are kept as abstract classes because we don't fix in this framework how the unit, drivers or stubs are generated. Thus, these classes are hot spots of the framework. At the time of instantiation of this framework one needs to refine these subclasses according to the application using the framework. The object diagram, that shows the relationships among all the types of classes identified during analysis and design, is shown in Figure 6.

4.3 Instantiation

We have instantiated this framework by adding three classes ***UnitLoader_Sub*** of *UnitLoader*, ***DriverLoader_Sub*** of *DriverLoader* and ***StubLoader_Sub*** of *StubLoader*. These sub classes allow us to provide unit, drivers and stubs (if required) manually using graphical user interface. Thus in this instantiation, we assume the application need is to provide these manually. In any other instantiation, some other method of providing a unit, drivers and stubs can be used for example as a result of some automation process that will generate them etc. Further one more class ***GUI*** is added that helps in implementation of these other classes added during instantiating. *GUI* class has a method that displays a string, passed it as a parameter, to the user. Thus, at the time of instantiation of this tight framework for 'EUT' only four additional classes were needed to be added.

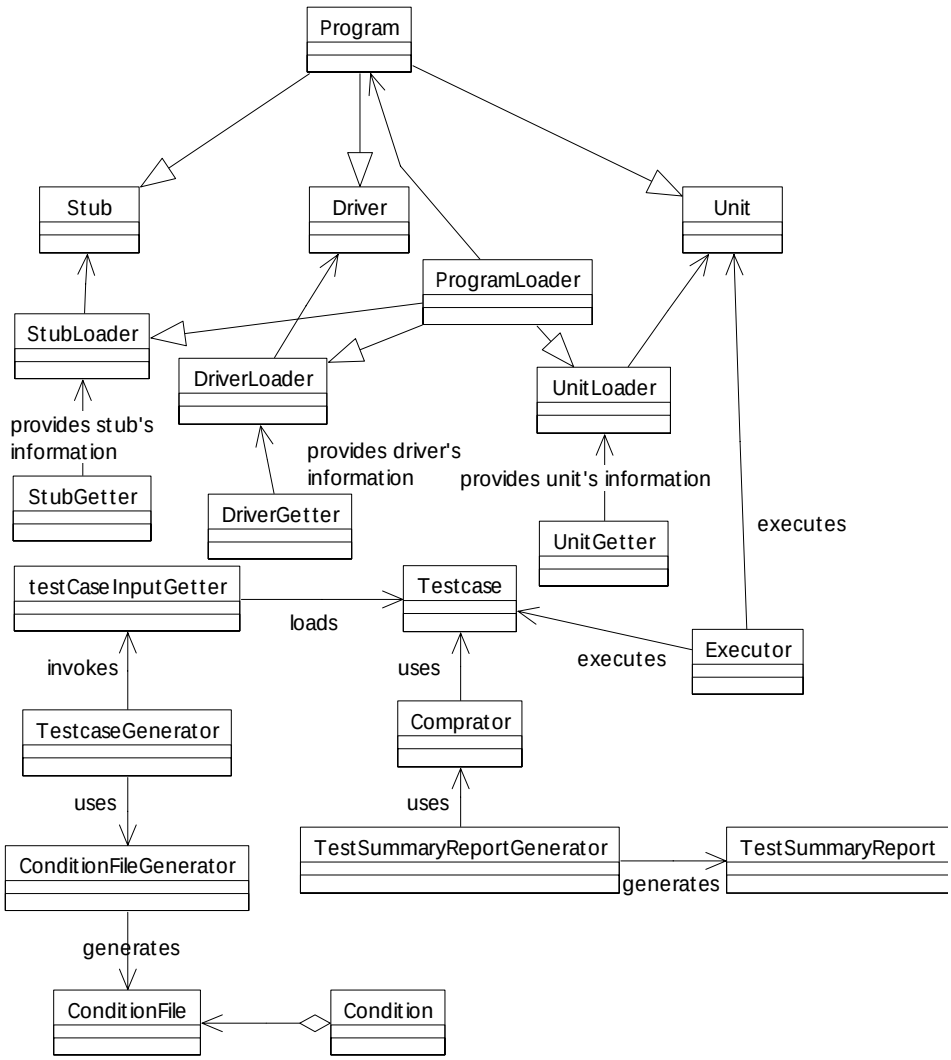


Figure 6. Object diagram of tight framework

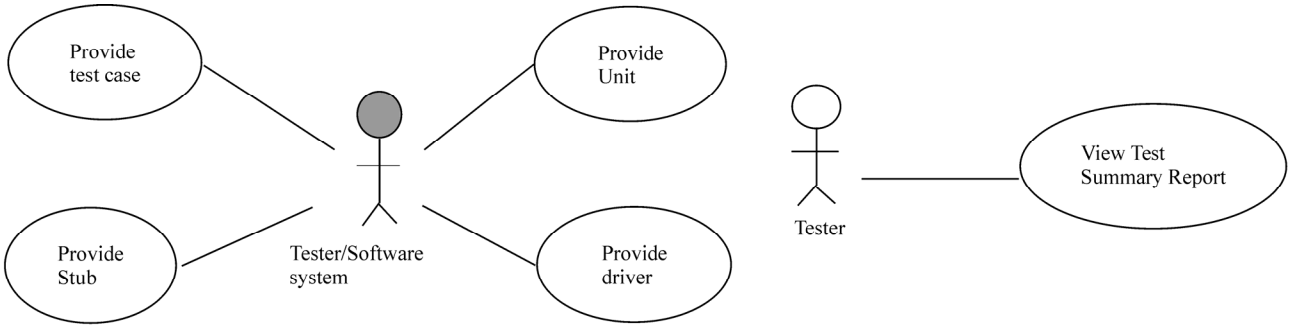


Figure 7. Use case diagram for loose framework

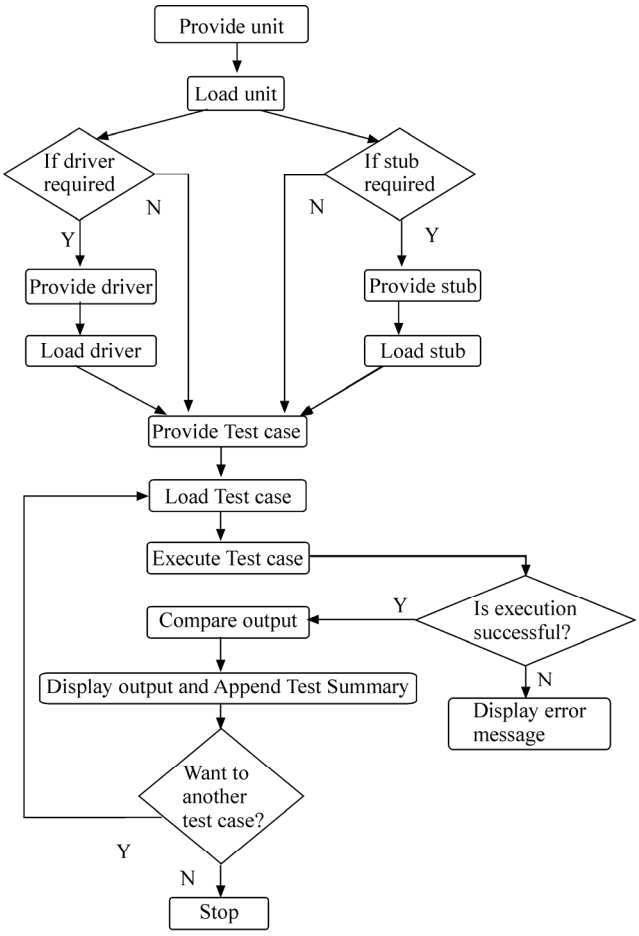


Figure 8. Activity diagram for loose framework

5. Loose Framework for 'EUT'

In the loose framework, for the same 'EUT' domain, we do not restrict the method of drivers, stubs or test case generation as we did in the above tight framework. However, in this loose framework the test oracle is also not fixed. That is, this framework would accept the drivers, stubs (if required for a unit) and test cases from somewhere else as the tight one did. These would be given to the framework either manually or generated automatically. This framework will accept and load these into the respective classes defined in the framework. This loose framework for 'EUT' does not fix the way of generating test cases, which is a big part of the framework. Thus, using this loose framework one can perform any type of structural testing. That is, the test case generation would also be application specific.

5.1 Analyzing the Loose Framework Requirements

By studying the problem statement, we specify the functionalities that the framework will support in the use case diagram in Figure 7. Gray ovals show functionalities that are needed to be customized and black ovals show functionalities that are prefixed in the framework. Unit, to be tested, drivers and stubs (if any) and test cases can be provided to the system either manually or can be loaded automatically using any software.

After analysis, here also, we get the same ***domain classes*** as we obtained in tight framework described above: ***Unit, Driver, Stub, Test case, and test summary report***. Only the domain class 'condition' that was identified during development of the tight framework, described above, is not a domain class for this framework. Rests of the classes are all same.

5.2 Designing the Loose Framework

To represent the dynamic behavior of a system that would be developed using this loose framework, we describe the following scenarios that explain how the system behaves when it performs some of its functions. The success scenario is as follows:

1) Unit is provided to the system.
2) If driver is needed, driver is provided to the system.
3) If stubs are needed, these are provided to the system.
4) Test data is given to the system.
5) Test oracle provides the correct output for a given test data.
6) System executes the test case and compares the actual output with the expected output parameter values.
7) System generates a test summary report.

Abnormal scenarios are as follows:

1) Unit is given to the system.
2) If driver is needed, driver is given to the system.
3) Test data is given to the system.
4) Test oracle provides the correct output for the test case.
5) System could not execute the test case and generates an error message "execution failed".

Similarly other abnormal scenarios would be if the unit and driver cannot be provided in the form used by the system. At that time system again generates error messages corresponding to the error occurred.

As we mentioned above, in this loose framework, we do not restrict the method of generating drivers, stubs or test case generation. Similarly the test oracle is also not fixed. That is, drivers and stubs (if required for a unit to be tested) and test cases can be provided manually or generated automatically and are supplied to this framework. This loose framework will accept and load these drivers, stubs and test cases into their respective classes defined in the framework. As the actual output would need to be compared with the expected output, these output variables would need to be stored in a file and then compared with the expected output. To redirect execution output of the unit it is required to add code in the unit for the purpose. Activity diagram (Figure 8) shows all the activities in this 'EUT' framework in brief.

Activity diagram hints some ***application logic classes***: UnitLoader, DriverLoader, StubLoader, TestcaseLoader, Executor, Comparator, TestSummaryReportGenerator. These classes are shown in Figure 9.

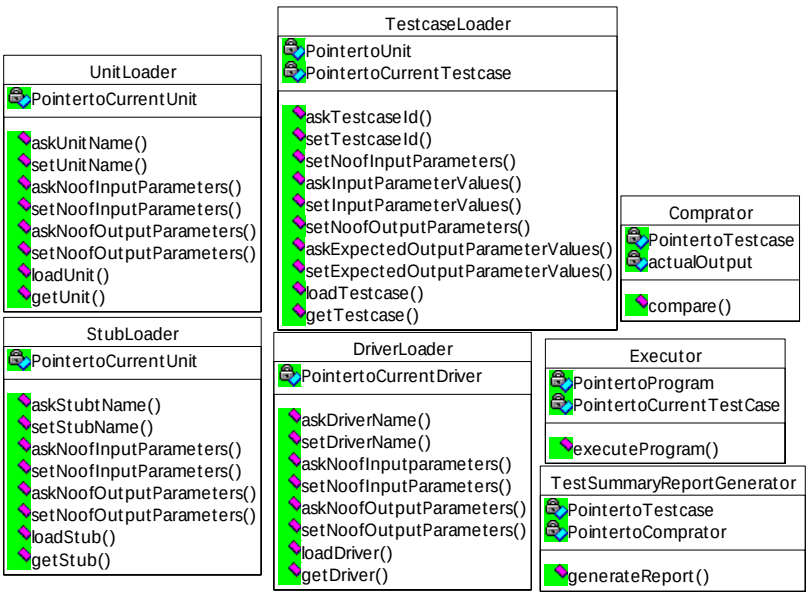


Figure 9. Control logic classes for loose framework

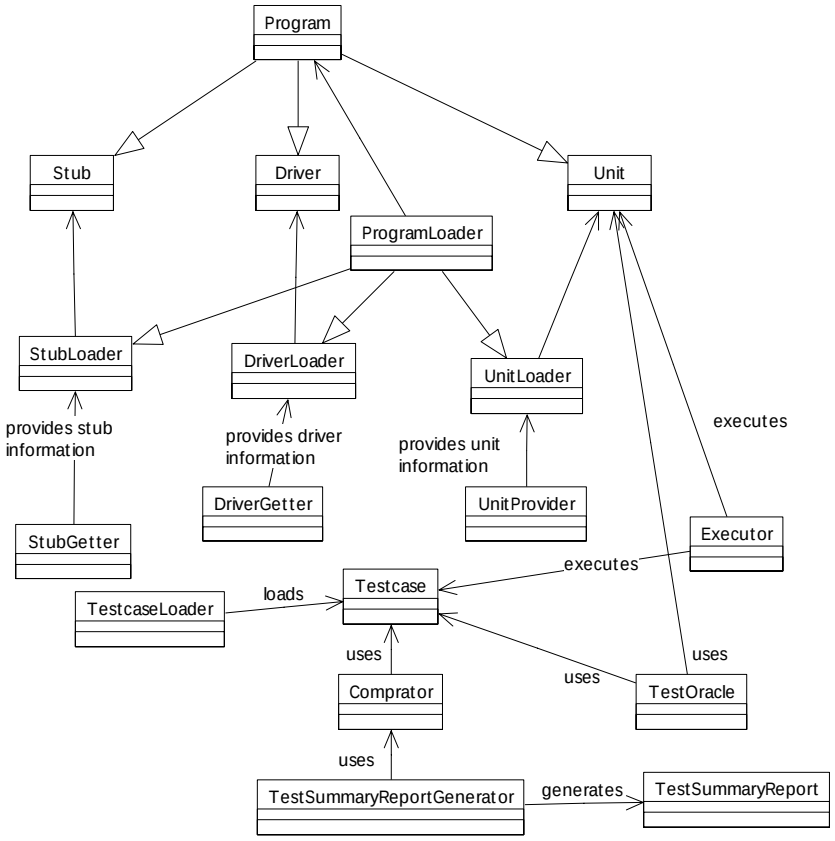


Figure 10. Object diagram of a loose framework

The user interface classes for this loose framework are same as they were in the above tight framework. Only the *TestCaseInputGetter* class is not included in this framework.

In this framework also, we take *Program* as an abstract class from which *Unit*, *Driver* and *Stub* classes would inherit properties and operations for the same reason as described in tight framework. Similarly, *ProgramLoader* is taken as an abstract class of which *UnitLoader*, *DriverLoader* and *StubLoader* are be implemented as sub classes for the same reason. Further, *UnitLoader*, *DriverLoader*, *StubLoader* and *TestcaseLoader* are also kept as abstract classes because we don't fix in this framework how the unit, drivers, stubs or test cases are generated. Thus, these classes are hot spots of the framework. At the time of instantiation of this framework one needs to refine these in subclass according to the application using the framework. After identifying the attributes and methods of these application-logic classes along with relationship among these and problem domain classes we get the object diagram (Figure 10).

5.3 Instantiation

We have instantiated this framework by adding five classes *UnitLoader_Sub* a sub class of *UnitLoader*, *DriverLoader_Sub* a sub class of *DriverLoader* and *StubLoader_Sub* a sub class of *StubLoader Test-CaseLoader_Sub* a sub class of the class *TestCaseLoader* and a class *GUI*. Except the TestCaseLoader_Sub rest of the classes have the same role and responsibilities that they have in the earlier described tight framework for 'EUT'. Since activities concerning with these classes were not fixed in the above tight framework also. Using this loose framework since the generation of test cases is not fixed in the framework there is a need to customize this activity also and thus the *TestCaseLoader_Sub* sub class is also added in the instantiation of this loose framework.

6. Observations

The objectives of the above case study are to obtain quantitative characteristics of frameworks for the purpose of comparing and understanding which framework (tight or loose) can better be reused in which scenario. The one main problem that we have encountered during this work is the lack of some good experimental data from real time environment that may help us to verify the proposed idea in an efficient manner.

Table 1. Comparison of tight and loose frameworks at code level

S.N.	Code Characteristics of frameworks	Tight framework	Loose framework
1.	Total number of methods	74	68
2.	Number of virtual functions	4	8
3.	Total Number of classes	18	13
4.	Number of abstract classes	4	5
5.	Size (Total number of non-commented lines of code)	624	485

A comparative table 1 shows different characteristics for loose and tight frameworks are drawn below.

By comparing frameworks at code level, we can answer the questions discussed above at some extent.

1) If a software framework has more number of abstract classes and virtual functions, the possibility of its reuse will be higher. As we know, the abstract classes and virtual functions give freedom (flexibility) to designers to instantiate a framework according to the requirements of a specific application. So such a framework can be reused in more number of applications. It can be shown from the table that loose framework for 'EUT' have 8 virtual functions and 5 abstract classes while tight framework has 4 virtual functions and 4 abstract classes. It shows the loose framework would be more reusable in terms of "number of reuses" because it can be used in more number of application developments than the tight one.

In a software framework, the abstract classes and virtual functions have to be customized at the time of instantiation of that framework. A designer/implementer would have to extend abstract classes and virtual functions according to the requirements of any specific application. If a software framework has more number of abstract classes and virtual functions then more effort would be needed to customize them at the time of instantiation. However, as autonomous, in terms of its service providing responsibilities, a framework would be that better contribution it can make in functioning of the system. It is shown in the table that loose framework for 'EUT' have 8 virtual functions and 5 abstract classes while tight framework has 4 virtual functions and 4 abstract classes. As identified in the definition of tight framework that it fixes the way of performing most of such activities in the framework itself, it is our conjecture that tight framework would be more reusable in terms of "ease of reuse". As shown in Figure 9 'TestcaseLoader' is an abstract class in the loose framework that is to be implemented according to the application specific requirements and hence requires extra effort. Whereas the tight framework has a concrete 'TestcaseGenerator' class that has all the implementation and hence it can be directly used in the application.

2) As given in the definition of *loose framework* that it is a framework that does not fix the way of performing most of the activities in the framework itself. During design of loose framework one need not to write semi-code for the portion that is not concretely defined in the framework. As shown in section 4, we did not fix the unit testing criteria in loose framework so we need not to develop code for generating test cases (based on any criteria) in this framework; only the interface that would connect the test case generator is needed to be developed. However, in the case of tight framework, one has to collect all the requirements for a specific application. Since, in tight framework, we fix the way of performing most of the activities so we have to write semi-code for all the activities. As shown in section 5, for the tight framework, where we fixed the testing criteria as condition testing, we required to develop whole code for generating test cases, satisfying this condition coverage criterion, as part of the framework itself. Based on our design and development experience regarding both type of frameworks, it is our conjecture that a loose framework would always be easier to develop than a tight one.

3) Unlike a loose framework, in a tight framework we fix most of the activities, so we have to write semi-code for them. It can be shown from the table that size of tight framework (total number of non-commented line of code) is 624 while size of loose framework is 485. And thus, the tight framework will be heavier, in terms of size, as compare to loose framework.

In case of tight framework, the interdependence among the different component of the framework would be more because the way of performing most of the activities are fixed. In order to understand/modify one component, one has to understand all the related components that make the tight framework more complex. In case of loose framework, different components are loosely coupled to each other. It is easy to understand/ modify one component, without understanding/modifying other components, in a loose framework because for each activity there would be perhaps different loose frameworks that fulfill the application need by interact-

ing with each other. As in our case, loose framework for 'EUT' interact with any of the test case generator framework and perform the unit testing. Thus whenever we need to modify in test case generator framework there is no need to understand the 'EUT' framework and vice-versa. Thus we can say a loose framework would always be less complex than a tight framework for the same domain. In short, we can say that the complexity of the tight framework would be high because several activities considered in that framework may result in tightly coupled implementation of them. However, the loose framework that contains only the abstract code would be less complex since no detailed implementation is there in its code.

One can consider the basic guiding principles for designing a software framework based on the above observations.

7. Conclusions

For some situations a tight framework would be better than a loose framework for same domain if the ways of performing the activities, fixed in the tight framework, are exactly those that are required in the needed application. In this paper, we suggested some scenarios, which type of framework would be more appropriate as compared to other one. These observations will be useful at the time of selection of frameworks. We have focused on limited parameters. Results would be more visible for industrial applications. One can extend this study by considering other quality criteria.

REFERENCES

[1] D. Roberts and R. Johnson, "Evolving frameworks: A pattern language for developing object-oriented frameworks," in Pattern Languages of Program Design 3. Addison-Wesley, Illinois, USA, 1997.

[2] J. Bosch, P. Molin, M. Mattsson, and P. Bengtsson, Object-Oriented Frameworks – Problems & Experiences, 1997.

[3] S. Sparks, K. Benner, C. Faris, and S. Consulting, "Managing object-oriented framework reuse," IEEE 1996, pp. 52–61, 1996.

[4] Y. J. Yang, S. Y. Kim, G. J. Choi, E. S. Cho, C. J. Kim, and S. D. Kim, "A UML-based object-oriented framework development methodology," Software Engineering Conference, Proceedings. 1998 Asia Pacific, pp. 211–218, 1998.

[5] IBM, "Building Object-Oriented Frameworks", http: //www.ibm.com/java/education/oobuilding/index.html.

[6] D. C. Schmidt, "Applying design patterns and frameworks to develop object-oriented communication software," Handbook of Programming Languages, Volume I, edited by Peter Salus, MacMillan Computer Publishing, 1997.

[7] M. E. Fayad and D. S. Hamu "Object-oriented enterprise frameworks: Make vs. buy decisions and guidelines for selection," The Communications of ACM, 1997.

[8] A. K. Tripathi and M. Gupta, "Risk analysis in reuse-oriented software development," International Journal of Information Technology and Management, Vol. 5, No. 1, pp. 52–65, 2006.

[9] P. Jalote, "An integrated approach to software engineering," ISBN 81-7319-702-4, Narosa. 2005.

Call for Papers

International Conference on Computational Intelligence and Software Engineering (CiSE)

计算智能与软件工程国际会议征文

December 10~12, 2010 Wuhan, China

http:// www.ciseng.org/2010

The 2nd International Conference on Computational Intelligence and Software Engineering (CiSE 2010) will be held on December 10~12, 2010 in Wuhan, China. This conference will cover issues in **computational intelligence, information security, multimedia and graphics technologies, and software engineering.** **The conference proceedings will be published by IEEE, and all papers accepted will be included in IEEE Xplore and indexed by EI Compendex.**

TOPICS

- Artificial Intelligence
- Automated Reasoning
- Autonomous Systems
- Cloud Computing
- Cluster Computing
- Cognitive Science
- Communication Networks and Protocols
- Computational Biology
- Computational Chemistry
- Computational Neuroscience
- Computational Physics
- Computer Graphics
- Data Modeling
- Database Mining
- Database Technology
- Evolvable Hardware
- Expert Systems
- Fuzzy Systems
- Geographical Information System
- Grid Computing
- Hardware Implementation
- Human-computer Interaction
- Hybrid Systems
- Image Processing
- Image Understanding
- Machine Learning
- Multi-Agent Systems
- Natural Neural Systems
- Neural Genetic Systems
- Neural-Fuzzy Systems
- Numerical Algorithms
- Operating Systems
- Pattern Recognition
- Programming Methodology
- Project Management
- Real Time Control
- Requirement Analysis
- Simulation and Modeling
- Software Engineering
- Software Maintenance
- Software Standards and Design
- Software Testing and Quality Control
- Symbolic Mathematics
- Technological Forecasting
- Visualization
- Web-based Services

IMPORTANT DATES

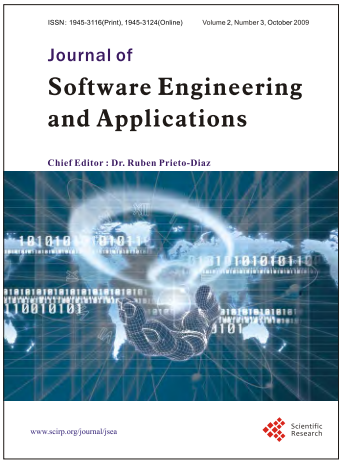
- Paper submission due: Jun. 21, 2010
- Acceptance notification: Aug. 10, 2010
- Conference: Dec. 10-12, 2010

CONTACT INFORMATION

Website:http://www.ciseng.org/2010

E-mail:info@ciseng.org

Call For Papers



Journal of Software Engineering and Applications (JSEA)

ISSN 1945-3116 (print) ISSN 1945-3124 (online)
www.scirp.org/journal/jsea

JSEA publishes four categories of original technical articles: papers, communications, reviews, and discussions. Papers are well-documented final reports of research projects. Communications are shorter and contain noteworthy items of technical interest or ideas required rapid publication. Reviews are synoptic papers on a subject of general interest, with ample literature references, and written for readers with widely varying background. Discussions on published reports, with author rebuttals, form the fourth category of JSEA publications.

Editor-in-Chief

Dr. Ruben Prieto-Diaz, Universidad Carlos III de Madrid, Spain

Subject Coverage

- Applications and Case Studies
- Artificial Intelligence Approaches to Software Engineering
- Automated Software Design and Synthesis
- Automated Software Specification
- Component-Based Software Engineering
- Computer-Supported Cooperative Work
- Software Design Methods
- Human-Computer Interaction
- Internet and Information Systems Development
- Knowledge Acquisition
- Multimedia and Hypermedia in Software Engineering
- Object-Oriented Technology
- Patterns and Frameworks
- Process and Workflow Management
- Programming Languages and Software Engineering
- Program Understanding Issues
- Reflection and Metadata Approaches
- Reliability and Fault Tolerance
- Requirements Engineering
- Reverse Engineering
- Security and Privacy
- Software Architecture
- Software Domain Modeling and Meta-Modeling
- Software Engineering Decision Support
- Software Maintenance and Evolution
- Software Process Modeling
- Software Reuse
- Software Testing
- System Applications and Experience
- Tutoring, Help and Documentation Systems

Notes for Prospective Authors

Submitted papers should not have been previously published nor be currently under consideration for publication elsewhere. All papers are refereed through a peer review process. For more details about the submissions, please access the website.

Website and E-Mail

Website: http://www.scirp.org/journal/jsea E-Mail: jsea@scirp.org

J. Software Engineering and Applications, 2009, 2
Published Online October 2009 in SciRes (http://www.SciRP.org/journal/jsea).

TABLE OF CONTENTS

Volume 2, Number 3 **October 2009**

9771945311006 04