International Journal of Smart Home
Vol.7, No.6 (2013), pp.111-122
http://dx.doi.org/10.14257/ijsh.2013.7.6.11

Research on Regression Testing Methods for Industry Applications

Lan Xiaowen

School of Information Engineering, Inner Mongolia University of Science &Technology, Baotou, China
lanxiaowenning@163.com

Abstract

This research discusses the problems about scope accuracy and quality control in application of regression testing in the engineering practice, and proposes a practical regression method, combing with change-impact-analysis, business rules model, cost risk assessment and test case management. This approach has already been applied to functionality testing of some core systems in our domestic financial institutions, and made great achievement.

Keywords: *Regression testing; Change-impact-analysis; Cost-risk-assessment; Industry application; Business rules*

1. Introduction

Software testing verifies whether software is up to standard by auditing software products and reviewing development activities. Popularizing test models specification greatly promotes the software testing process and improves the quality of the software. However, the traditional test model has focused attention on the software development process testing, not placing enough emphasis on regression testing, and it does not solve the issue on the accumulation of knowledge in the industrial testing application.

With the wide application of IT in various industries, software systems have became extremely important. The reliability of the system is playing plays a key supporting role for the application business development. Development and maintenance are always accompanied the entire life cycle of application systems. As a result, there is a growing demand for regression testing.

Industry application regression testing has its inherent characteristics. The first is business-related. Application changes mainly come from business development. Because new business and old business are inextricably linked, which bring great difficulties to define the scope of regression testing. The second is iteration. Regression testing is an iterative process, a new round of testing has a great similarity with the pre-test, so how to reuse historical accumulation of test resources, and improve testing automation to efficiently complete testing are worthy of further study.

The present researches are mainly focused on unit testing, which aims at reusing of functional functions and objects of the class-level, so they can efficiently complete unit regression testing. However, there are fewer researches on regression testing of functions, which are closely related to industry applications. Due to different industry backgrounds and business rules, it is very difficult for testing operators to define the impacts of business functions, which are directly or indirectly caused by the system changes. Furthermore, the cases for functional test are closely related to the business logic, as a result, most cases can not be reused after the changing of system, which

ISSN: 1975-4094 IJSH

leads to lower efficiency of regression testing automation. Directed at these problems above, this research designs a set of regression testing methods, with the consideration of technology of change-impact-analysis, business rules model, cost- risk-assessment and test case management.

2. Software Testing Model and Regression Testing

A software test is regarded as a basis for the work of software testing. It is a framework of the entire process of software testing, and it not only can explicitly display the whole process of software testing, but also clearly regulate the main activities and tasks to complete.

The classic waterfall model divides software development process into several stages, including requirements analysis, directions of standard, outline design, detailed design and coding. Barry W. Boehm proposed V-model, which responses to the traditional waterfall development model at every phase. After the completion of the coding, the model fulfills for unit testing, integration testing, system testing and acceptance testing.

Paul Herzlich proposed W-model in 1993. Compared with V-model, it adds synchronization verification and validation at every stage. W-model emphasizes that testing should accompany the entire software development cycle, strengthen synchronous testing of demand and design to find out where the problem lies as soon as possible.

Marik summed up some shortcomings of the V-model, and presented X-testing model. The model does not fully describe a whole life cycle of software development, but it aims at the iteration of testing and development, emphasizing the exploratory testing.

Pre model emphasizes the tests conducted on each of the deliverables, and it consists of technical tests and acceptance tests. Technical testing is mainly aimed at developing code testing, while acceptance testing is to verify users' criteria before accepting the delivery systems. The model combines testing execution and development together, embodying the way of coding-testing,-coding-testing during the development phase.

The common testing models above are development-based; stressing how to infiltrate testing process into development process to improve software development quality. However, in the Software life cycle, after the development, maintenance and upgrades cost more time and money, therefore, after the system has been up and running, the losses caused by defects due to software changes will be significantly more than the cost in its development stage.

Regression testing is important for a software testing, and it focuses on variations occur in the software life cycle, and the resulting quality of the software may give side effects. Regression testing is used to monitor the software changes, and timely feedback from the consequences of change. When the software is modified, and certain aspects of the software configuration (programs, documents, or data) have also been changed, regression testing is used to guarantee these changes not to introduce new bugs or generate other code errors.

Regression testing may become very large during the process of integration tests. Thus, it is unpractical and less efficient to rerun all the tests for each modification

Different from the testing in development stage of testing, in most cases, regression testing repeats the functions which have been successfully tested in the past, so it has a higher requirement about the reused test cases. Therefore, the automation of generating

test cases can significantly improve the quality and efficiency of large-scale software regression testing.

3. Problems and Countermeasures for Regression Testing in Industry Application

3.1. Problems Faced

There are typically two major problems for regression testing of large-scale business systems. Firstly, regression test coverage can not be accurately defined with the changes of system; Secondly, the number of test cases expands dramatically with the combination of parameters, so it is unable to complete regression testing of the minimum coverage requirements within the determined period of time at a reasonable cost.

Automated functional testing tools are frequently introduced in the testing of large business systems. These tools provide a basic means of testing, but t automatic function test management framework is not available, which leads to the fact that automated functional tests are often unable to be effectively implemented and carried out. The root cause is that functional testing is based on business, with a strong industry relevance, but automated functional testing tools are not related to business, so it cannot automatically adapt to the specific business needs of each industry, and it requires a lot of human intervention during the implementation of the testing process, and the results are often difficult to meet people's expectations.

Regression testing of large-scale business systems tends to be restrained by the deadline and budget constraints, and engineering properties of the test determine that it is impossible to achieve completely as it describe in theory. With the limited time and resources, in order to make more rational arrangements for testing, a decision-making mechanism is of great need in testing planning phase to constraints resources (time, manpower, budget) based on the premise of risk assessment and (test) cost estimation for decision making.

3.2. Methodology

The previously mentioned test models are relying on software development process, so there is no practical implementation approach for regression testing. Different from the unit testing, integration testing and performance testing in development process, regression testing repeatedly emphasizes accumulation, which can be completed through the structure and the business rules modeling methods, so that the cycle of regression testing can proceed.

In order to cope with problems that large-scale business systems face in regression testing, bridge the gap between business systems and automated testing tools, we propose and establish a supporting platform for regression test decision. The platform can be modified according to the system changes, accurately pinpointing the scope of change, and producing the corresponding test point under the application of the related business rules. And then, with the help of the risk assessment model, implementation plan can be determined, and test cases can be reused as much as possible. Finally, automatic regression testing complete will be fulfilled.

To build a supporting platform of regression testing for decision-making, at first, you need to scan and analyze the source code of the core business systems, and set up an application description model; meanwhile, a bank of expert knowledge of the industry should be established to collect and refine business information. And then, a model of

business rules should be established to express business information. Finally, risk assessment model will be established, according to industry application and the characteristics of test implementation.

In regression testing, reusing of used cases can greatly improve test efficiency, and reduce time and duplication of effort. Therefore, there is a huge test case library at the supportive platform. It indexes all the used cases in catalogue to associate the specific cases with related businesses, and facilities the reference of cost-assessment model and the automatic generation of the test scripts.

If business systems change with the modification of demands, and with the changes of system maintenance and other reasons; if new versions of the software are produced by the development department, implementation steps regression testing of are as follows:

(1) Scan and analyze the source codes in the new version, and conduct analysis of changes bases on the application model, automatic identify system changes;

(2) Analysis of change impacts analysis accurately pointed out the scopes of functional business directly or indirectly influenced by a change of version.

(3) With the application of business rules, the regression test ranges are determined by experts and analysts

(4) Test suite is generated in the assessment model of cost and risk, and it will be compressed with optimization algorithm;

(5) Complete automatic testing by refusing used test cases in the library or developing new cases.

The above process can be illustrated in Figure 1.

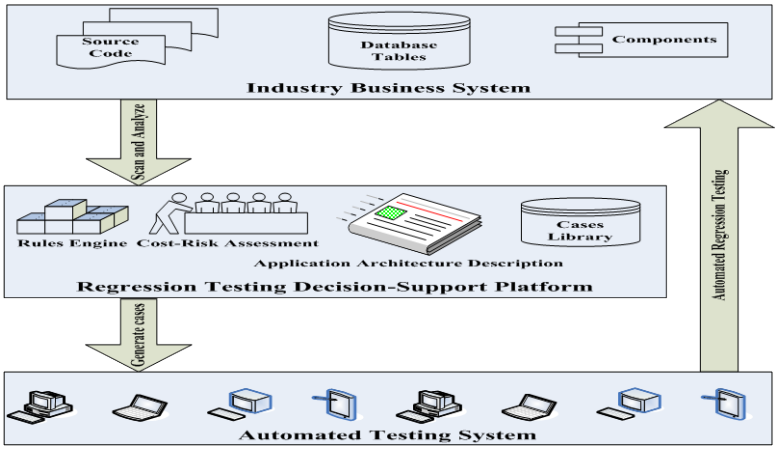

Figure 1. Implementation Process of Functional Regression Testing

After the completion of a regression testing, there is a need to classify the newly developed test cases, classified according to the degree of reusable test cases, and save them in the library. With the introduction of new services, business rules need to be further improved; meanwhile the relevant parameters in cost-risk assessment model can be adjusted and optimized according to the test results.

With the advancement of the software life cycle, business systems development and repeated iterations of regression testing, the expression of business rules gradually

become improved, test cases will be more sufficient, and implementation of regression testing will become more and more efficient.

4. Regression Testing Methods for Industry-oriented Application

Building a decision-support platform of regression testing provides a viable solution to industrial applications of regression testing. The construction involves models of business rules, application description model, change-impact-analysis, cost-risk-assessment, and test case management.

4.1. Extraction and Loading of Business Rules

Business rules are defined as constraints and norms for business structure and operation. They are important resources for enterprise business operations and management decisions.

In the age of manual testing, test requirements analysis is based on the analyst's personal understanding of applicable rules, so the analysis of the results is different from one to another. Furthermore, because the rules are not explicitly expressed, there are some problems on integrity and consistency of the rules in practice. At the same time, those rules are the accumulation of a senior analyst's the experience, with time passing by, it formed an industry tradition of manual testing times.

Business rules should be managed by the rule-based system, thereby separating application logic from the business process logic of application system. Rules engine is an embedded component in an application program. Its task is to test and compare the object data which have been submitted by the rule with the original rules, activate rules that meet the current state of the data, and trigger corresponding actions in the application program, according to the rules declared in the executive logic.

To build business rules model supported by regression testing is to inherit the accumulated knowledge of senior analysts, so that there is an explicit expression for the actually used rules. On this basis, combining test theories and rules integration and optimization algorithms with the case, we can establish a generation system, which is not less efficient than an average level of case generation system in manual test. Figure 2 depicts the general rule of extraction source.

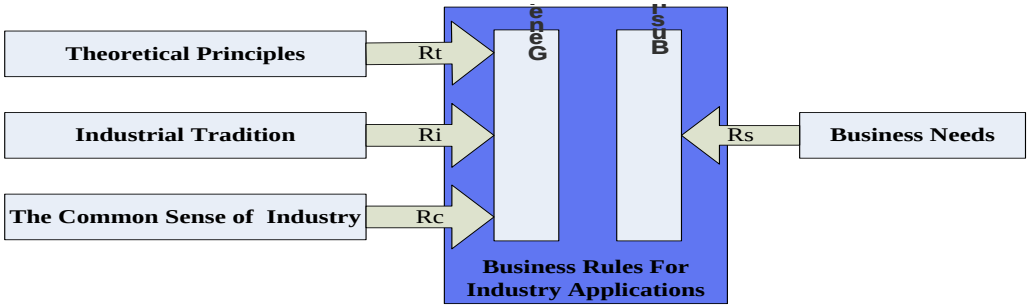

Figure 2. Extraction Sources of Business Rules for Industry Applications

The sources of business rules generally include:

(1) Rules derived from business needs (Rs)

(2) Rules derived from the theoretical testing principles (Rt)

(3) Rules from the industrial tradition (Ri)

(4) Rules from the common sense of the industry (Rc)

The basis of business rules model is the accumulation of a series of designing rules, industry standards, and special constraints from operations (such as legality standards of input elements) in manual test cases. Business rules model is used to express these rules in manual testing age, and establish a structure of rule engine which can be loaded rules. With these rules, a basic template case can be generated in the supportive system of decision-making for a specific business process.

Loading rules is to add a rule to the rule base. The key point is how to express the applicable conditions and specify optimization algorithms.

The expression of business rules is specific, and its basic form is If (applicable conditions of rules) Then op, among which Op both means generation of test points and case algorithms.

For a target system, it is impossible to exhaust all possibilities, it can only advance progressively. Therefore, manual addition should be allowed, and it is regarded as a learning process for business rule model.

4.2 Application Description Model and Analysis of Impacts

For software regression testing, it is essential to determine the scope of the impact of changes. The scope of impacts of changes is exact target range of regression testing. If you can not determine the scope of the change, then the theory would have to re-test the whole system again, for large systems, this cost is intolerable.

Two prerequisites for change-impact-analysis

(1) There is a structure（impact criteria） to describe changes, such as < object, process, type >, and to express each change, so that (many) changes become a set of impact criteria.

(2) There is a description of the internal structure of the system, in other words, there must be sufficient models to describe the overall system structure and data.

At decision-support platform, description model is to abstract modeling for large business systems, and it acts as a basis for variance analysis and expresses business scenarios. It mainly accomplishes two descriptions: One is description of system architecture. The materials of this description are from source codes of system, including elements of classes, functions, components, libraries, and tables. Work conducted by the analysis system of source codes is a preparation for this resource; The other is business description, including business function points and business processes, its material are from the business system requirements and development standards. It ultimately associates the system architecture with software business, providing the basis to determine the scope.

Source code analysis is to extract the needed information from the source codes for application structure. Studies show that for object-oriented systems, dependency analysis method can express the intrinsic link applications. Based on class and package diagrams and other graphic expressions, you can build classes, subsystems dependencies and generated classes, subsystems dependency sets [1]. In addition, the use of regression testing strategy of class member’s firewall can extract class hierarchy from the source program. With the class of Analytic Hierarchy Process (CHA), an

accurate dependency graph [2, 3] of the class members can be obtained according to class hierarchy structure.

For industrial applications, tools for the source code analysis also need to extract some relationships of business process and component, component and component, component and class hierarchy, components and associated database table. The premise of this work is that source code analysis tools can properly describe morphology and grammar of the target system codes, extract the target system objects, processes, relationships between data, and identify the relationships between database tables through the analysis of the embedded sql statements.

For object-oriented systems, effects of the changes can be alleged to the objects and methods. Michelle l. Lee (1998) discussed a method. He defined change descriptor for object-oriented system as <C, CM, CT>, where C represents the proposed change of class; CM represents proposed changes among members, CT stands for change types. Lee proved in form that when calculating a group changes of effects, we can firstly initiate the affected class collection (ICS) into changed class collection (Ci), and initiate each affected member of function collection (IFS (Ci)) and affected members of data collection (IDS (Ci)), then conduct stack generation.

At decision-support platform of regression testing, we make use of analysis tools derived from Javacc to analyze business system source codes, and generate the corresponding syntax tree. During this process, the embedded database operation statements are analyzed lexically, and the nodes of syntax tree get association with the related database tables. With this syntax tree, we can build the various connections between each module within the system as well as the dependencies between database tables, and get description of the application structure.

Application model aims at description of internal business system structure at a specific period, it should correspond with the version of business system software. When a business system changes, a corresponding new version of the application description model will be generated by scanning the analysis. Change-impact-analysis of the two versions is to analyze the differences of the two corresponding models of the two versions, and determine the range of changes when the software modifies. Meanwhile, the affected but the unmodified portions are clarified and the target ranges for regression testing are obtained. The process is showed in Figure 3.

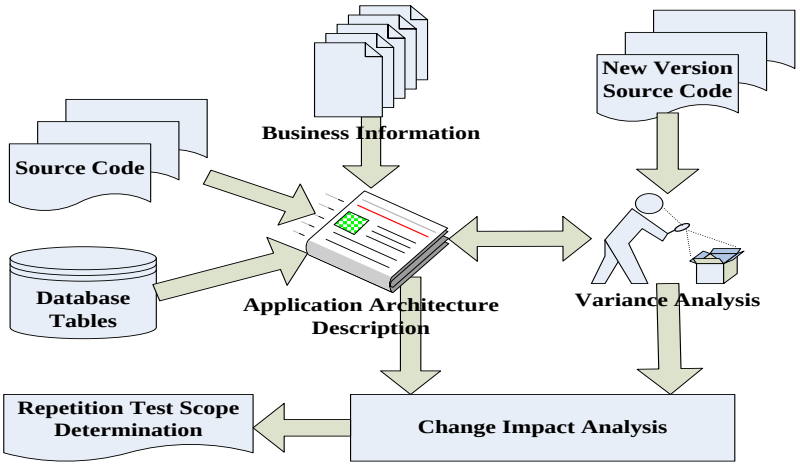

Figure 3. Determination of the Target Ranges for System Regression Testing

4.3. Cost-risk-assessment

An effective testing is not necessarily the most comprehensive, but it should be the most targeted. The purpose of testing is to reduce risk, so the process of risk analysis is cruicial.

In software engineering, a problem that has been plaguing project managers is how to plan and organize testing more rationally within the limited budget of time and resources. A famous economist and sociologist Vince Weafer·BaiRuitu presented "Eighty/twenty" principle to guarantee the quality of the software, where the majority (80%) of the limited resources are assigned to the "vital few" (20%). This is because that at any particular group, important factors usually account for only a small number, while the less important factors account for most, so as long as we can control the few factors of importance, we can control the whole group. In the field of software testing, how to effectively identify "critical few" functional modules, is essential to improve testing efficiency. Testing strategy based on risk analysis is a good solution to the problem.

Regression testing in engineering practice is often subject to on-line time, the implementation of the budget, labor inputs and business-critical constraints of test objects. Therefore, in the implementation process, a scientific test planning mechanisms is of great requirement to ensure that all conditions are met the prerequisite constraints.

Boehm defines “risk” with the formula RE=P (UO) *L (UO) , among which RE refers to risk or the effects of risk; P (UO) stands for the probability of some dissatisfying results, L(UO) is destructive of degree [4] of the bad results.

Therefore, the value of Re of the test object can be expressed with the multiplying of error probability and the cost of errors:

$$Re(f)=p(f)\cdot L(f)$$

Determining the probability of the risk occurring to the test objects and the corresponding loss is of the process of risk assessment phase. The Re of each test object can be obtained by multiplying the probability of the risk occurring and the loss caused by the risk.

Notably, the risk assessment itself is a subjective evaluation, so its results are of a certain amount of uncertainty. This uncertainty depends on the accuracy of measurement and measurement systems which will affect further decisions.

To facilitate quantitative estimation, the risk assessment model applies the method of grading costs to express the cost of error and risk probability. Risk probability involves maturity of the model and the expected errors of the developers. The basis of the expected errors is the recorded defect rate (bug per k lines). When the record is absent, the maturity of all the developers will be regarded to be high, then the individual maturity score changes with the relative value of individual defects Rate (br) to average defect rate (abr) and the trend of individual defect rate (tbr).

According to the estimation of risk value, the minimum test depth of the object to be tested can be determined. With test depth requirements, choosing a simple algorithm for the appropriate strength tests is of the first priority, and then the number of required cases can be drawn.

Because a large number of cases can be reused in the regression test, calculating the cost of implementation requires distinction between the tests cased generated automatically and the cases completed manually. Calculating with different weights eventually concluded that the implementation costs of testing process, as shown in Figure 4. Implementation costs can be expressed by man-hours input. In light of the

comparison between the cost and time, whether the testing cost is in the acceptable range can be determined. If it is acceptable, the final test suite can be generated with the corresponding simple algorithm. If implementation cost is unacceptable or if there is a surplus, the simple algorithm should be adapted, and the testing strength will be reduced or increased, recalculation of the implementation cost has to be conducted until it meets budget requirements.

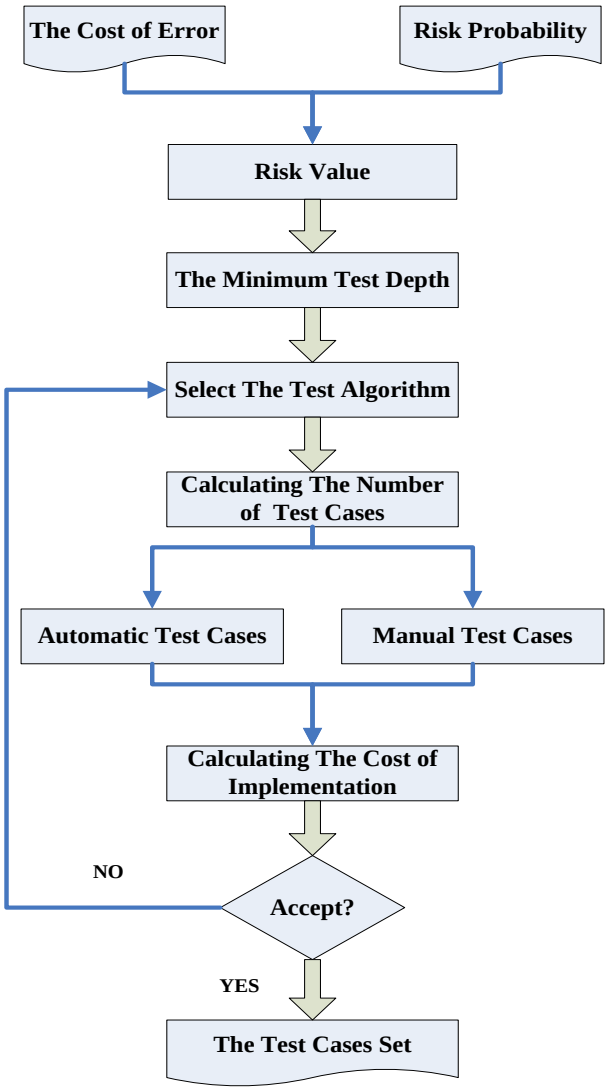

Figure 4. Value Assessment of Cost and Risk

4.4. Test Case Management and Report Management

At the supportive platform with regression test for decision-making, the use case is generated after the cost-risk-assessment, and the rule engine induces the underlying use case templates and data, generating test cases based on business descriptions, and then produce automation test scripts by the engine of interpreted cases. There are three ways to generate used-cases: reusing cases directly, partly reusing, and new development. Direct reusing is to use the archive cases accumulated in the past implementation; part reusing means that some adjustment should be done due to changes in the business

environment or test the constraint conditions; new cases development is the result of t the newly-developed business, so there is no archive case to reuse and it requires specialized design.

Regression testing is quality inspection and evaluation of a business system with different versions, and it is an iterative process, in which abound recurring of the same business occur and reusing of test cases is particularly important.

The bank of test cases is an important part of regression testing decision support platform; it can be applied to risk assessment models as well as to the cases of inducing business rules. Cases arise from case management process, which is in the charge of administrators, test designer, and test engineers. After implementation of the regression tests, the test designer is responsible for the analysis of the test case to identify reusability of the use case by distinguish the dependent elements based on their reliance on environment. After the analysis, the configuration manager classifies the t case files (including reusable and un-reusable cases). Those reusable cases are filed separately, creating reusable case files; and they are added into risk assessment models as well as into the business rule engine to enrich the accumulation of cases, and do preparations for the next round of regression testing.

Regression testing is accompanying the whole process life cycle of the application system. During the whole process, it will generate a lot of test reports, which contains a great deal of knowledge and experience. At decision-support platform, a catalog retrieval system is established based on document management, content management, resource management, realizing the process of exploring of "testing knowledge" and enhancement of services and value. The catalog retrieval system of test reports aims at exploration of internal testing knowledge. It is an accumulation of standard test documents, information and resources. It turns the test knowledge of various roles in the internal working process into advanced testing knowledge. With repeated conduction of the above process, a more adhesive system of testing knowledge will be completed to professional services.

5. Conclusion

Software tests, especially software represented by regression testing, it accompanies the whole life cycle of industrial application system. This paper presents a regression testing method for industry-oriented applications to solve issues, such as the low degree of automation of large-scale business systems and difficulty of defining test coverage. Benefiting from this method, decision-support platform of regression testing has been applied to a number of large financial systems and made great achievements.

Acknowledgments

This research was supported by Natural Science Foundation of General Projects of Inner Mongolia Autonomous Region, China (2013MS0915) and Innovation Foundation Projects of Inner Mongolia University of Science&Technology(2012NCL049).

References

[1] H. Shunren, J. Ximing and Z. Dengyi, “Research of Regression Testing on object-oriented System”, Journal of Chongqing Institute of technology, **(2005)** May.
[2] L. Haihong, M. Li and Z. Dafang, “Two Approaches on Chang-Impact Analysis of object-oriented programs”, Computer Engineering and Science, **(2005)** May.

[3] X. Haibo, Z. Dafang, M. Li and T. Xiaolan, "Research on Changes and impacts of object-oriented software", Science, Technology and Engineering, **(2005)** October.
[4] B. Boehm, "Software Risk Management: Principles and Practices", IEEE Software, **(1991)** January.
[5] C. Lingzhi and S., "Boa source code analysis and its application in embedded system", Computer and Digital Engineering, vol. 6, no. 33, **(2005)**, pp. 10-12.
[6] F. Neyret, "Qualitative simulation of convective clouds formation and evolution", Proceedings of the 8th Eurographics Workshop on Computer Animation and Simulation, **(1997)**, pp. 113-124.
[7] K. Nagel and E. Rasehke, "Self-organizing criticality in cloud formation?", Physica A, **(1992)**, pp. 519-5031.
[8] Y. Dobashi, T. Nishita and T. Okita, "Animation of Clouds Using Cellular Automata", Proceedings of Computer Graphics and Imaging'98, **(1998)**, pp. 251-256.
[9] Q. Bo and L.V. Tao, "Real-time dynamic cloud modeling and rendering", Computer Graphics, Imaging and Vision: New Trends, **(2005)**, pp. 285-290.
[10] J. Xu, C. Yang, J. Zhao and L. Wu, "Fast Modeling of Realistic Clouds", Computer Network and Multimedia Technology, **(2009)**, pp. 1-4.
[11] E. M. Upchurch and S. K. Semwal, "Dynamic Cloud Simulation Using Cellular Automata and Texture Splatting", Proceedings of the 2010 Summer Computer Simulation Conference, **(2010)**, pp. 270-277.
[12] T. Yanagita and K. Kaneko, "Coupled map lattice model for convection", Physics Letters A, vol. 175, pp. 415-420.
[13] R. Miyazaki, S. Yoshida, Y. Dobashi and T. Nishita, "A method for modeling clouds based on atmospheric fluid dynamics", Proceedings of Pacific Graphics, **(2001)**, pp. 363-372.
[14] Y. Dobashi, T. Nishita, R. Miyazaki and S. Yoshida, "Modeling and Dynamics of Clouds Using a Coupled Map Lattice", Proc. Siggraph 2001 Technical Sketches, Los Angeles (USA), **(2001)** August, pp. 229.
[15] J. T. Kajiya and B. P. Herzen, "Ray Tracing Volume Densities", Computer Graphics, vol. 18, no. 3, **(1984)**, pp. 165-174.
[16] D. Overby, Z. Melek and J. Keyser, "Interactive Physically-Based Cloud Simulation", Proceedings of Pacific Graphics 2002, **(2002)**, pp. 469-470.
[17] R. Miyazaki, Y. Dobashi and T. Nishita, "Simulation of Cumuliform Clouds Based on Computational Fluid Dynamics", Proceedings of Eurographics 2002 short presentation, **(2002)**, pp. 405-410.
[18] R. Mizuono, Y. Dobashi, B. Y. Chen and T. Nishita, "Physics Motivated Modeling of Volcanic Clouds as a Two Fluid Model", Proceedings of Pacific Graphics'03, **(2003)**, pp. 434-439.
[19] M. J. Harris, "Real-time cloud simulation and rendering", Chapel Hill: University of North Carolina, **(2003)**.
[20] Blinn, James F., Light Reflection Functions for Simulation of Clouds and Dusty Surfaces[C]. Computer Graphics (SIGGRAPH '82 Proceedings), pp. 21-29, August 1982.

Author

Lan Xiaowen received the BS in computer science from Inner Mongolia University,China,in 2001,and received the MS in computer science from Chengdu University of Technology,China,in 2008.Currently,his research interests include embeded system and scheduling techniques and parallel algorithms for clusters, and also multi-core processors and numerical simulation.