A Case to Do Empirical Study Using Educational Projects

Mingyang Gu
Norwegian University of Science and Technology, Trondheim, Norway

mingyang@idi.ntnu.no

Abstract

Nowadays, many empirical studies are carried out based on educational projects. In these cases, instructors put forward requirements on educational projects from the perspective of education, and researchers also provide their requirements from the perspective of empirical study. How to design educational projects successfully from both educational perspective and research perspective becomes a practical challenge. In this paper, we present a successful case to resolve this problem: we designed two educational projects in a software architecture course, based on which an empirical study about creativity in software development was carried out. Our methods to design these educational projects can be reused for other software engineering courses associated with empirical study.

Keywords: Educational project, empirical study, software architecture, software engineering education

Introduction

To provide students with the project-based software engineering course is accepted in software engineering education community and some good examples have been reported (Andersen, Conradi, Krogstie, Sindre, & Sølvberg, 1994). In order to meet different educational goals, instructors put forward various requirements on these educational projects.

One important issue which gets more and more attention currently is about the industrial relevance, which asks the students to experience educational projects in real industrial context or simulated context (Lethbridge, 1998).

Shaw gave us a roadmap in the field of software engineering education (Shaw, 2000). In this paper, she described current status and challenges about software engineering education, and suggested keeping education current in the face of rapid change in software engineering. She also argued that educational institutions should teach their students adapting to rapid changing skills used in industrials. So, educational projects should adopt the state-of-the-art techniques and processes used in industrials currently.

Besides teaching isolated topics used in software development, instructors also need to integrate these isolated topics into an integrated development process (Schneider, & Johnston, 2003), in which students can experience how isolated topics constitute one integrated development process.

Lawrence Bernstein argued that computer science students are typically technology-oriented and process-averse, so they always fight against process disciplines when they take any software engineering course. And when they finally understand the importance of these disciplines, they have to re-learn them again (Bernstein, & Klappholz, 2001). And he also provided a method to solve this problem: at the beginning of one course, students are forced to live through a project in which some key disciplines are ignored designedly, and then students will be shocked into an awareness of the importance of these special disciplines and motivated to learn them. Specially, Royce argued that university education should organize processes to make students recognize the importance of software architecture and where to fit software architecture into the software life-cycle (Royce, Boehm, & Druffel, 1994).

Summarizing the issues above, instructors should provide students with educational projects in real industrial context or simulated industrial context, in which students can experience how isolated topics can be integrated into a development process, and they also need design a process elaborately to teach students the importance of involved software engineering disciplines at the beginning of one course.

On the other hand, researchers in software engineering community have realized the necessity of empirical studies, and argued to carry out empirical studies to test methods, processes and tools before they are applied to industrial context. However, empirical studies in industrial settings require a lot of money and time, and may cause many technical or manageable risks, so they need to be carefully planned and implemented. Some empirical studies are even impossible or prohibitive in industrial setting, for example, the controlled experiment to compare the applicability of several development methods through implementing one project using each of these different development methods.

Since project-based software engineering courses have provided software development processes, researchers have begun to design and carry out empirical studies using educational projects (Carver, Jaccheri, Mrasca, & Shull, 2003).

So, here come two types of requirements: pedagogical requirements and research requirements, when we design educational projects based on which empirical studies are carried out. And not all of these two types of requirements are consistent, so how should we balance pedagogical requirements and research requirements to make an educational project successful from both an educational viewpoint and an empirical viewpoint becomes a practical challenge.

Jeffrey Carver etc. have touched this challenge in (Carver, Jaccheri, Morasca, & Shull, 2003), in which they identified four roles in empirical study using students as subjects: students, instructors, researchers and industrial, and listed the benefits and costs to each of them. And all these rolls should be taken into account when designing educational projects.

In spring of 2003, we designed and carried out two educational projects: KWIC and Mobile Robot in a software architecture course. Based on these two projects, we carried out an empirical research about creativity in software development. In this paper, we will present how we identified and balanced pedagogical requirements and research requirements to design these two educational projects. The students' evaluations to these projects were extracted to illustrate that our projects design was successful from both the pedagogical perspective and research perspective. In our design process, we adopted an attitude that if research requirements are conflict with pedagogical requirements, we leaned to pedagogical requirements (Cooper, & Schindler, 2003).

The rest of the paper is structured as following: pedagogical requirements and research requirements are identified in section 2 and section 3 respectively; in section 4, we combine these two types of requirements, and design two educational projects; in section 5, students' evaluations to

these two projects are analyzed to illustrate that both pedagogical requirements and research requirements are satisfied in these two projects; Our conclusions are drown in section 6.

Pedagogical Requirement for Software Architecture Education

Software architecture deals with the transition from requirement specification to software design. The architecture of a software system is composed by a set of views which are used to express the system from different perspectives (viewpoint). Different stakeholders can analyze the architecture from different perspectives to explore the defects hidden in the architecture or evaluate different architectures to select the most appropriate one to implement.

In spring of 2003, we gave a project-based software architecture course. Guided by the literature study about software engineering education discussed in introduction section, we identified the following pedagogical requirements (PR) on educational projects:

PR1: In order to motivate students to study the knowledge of software architecture purposefully, educational projects should teach them the important and the difficult topics of software architecture at the beginning of this course (before the operational methods of software architecture are taught).

PR2: Educational projects should give students the chance to experience how to describe software architecture accurately, and how to assess software architecture effectively.

PR3: Educational projects should provide students with a chance to experience how software architecture is integrated into the whole software development.

PR4: In order to improve students' interests and educational effects, educational projects should provide multiple software architecture designs, architecture assessments and system implementations so students can compare them and compete to each other.

PR5: Educational projects should provide an industrial context or a simulated industrial context.

PR6: Educational projects should use the state-of-the-art techniques used in industrial context currently.

PR7: The project setting should not use too much time on the topics which have little to do with educational issues.

Research Requirements from Empirical Study about Creativity in Software Development

Human creativity is thought as the source to resolve complex problem (Glass, 1995) or create innovative products (Humphrey, 1987), and several researchers have point out the importance of creativity in software development (Glass, 1995), (Dybå, 2000), (Winograd, 1996). Up to now, however, most of studies about creativity in software development are qualitative or prescriptive. In order to start empirical study in this field, we identified the following three motivational questions (MQ) as following:

MQ1: Which phases in software development are perceived to include more creative work compared with discipline-based work?

MQ2: Can UML-based documentation increase or decrease developers' perception about the amount of creative work?

MQ3: Can more creative work accelerate or decelerate development speed?

Here, we define creative work as the work (process) that is perceived to be helpful to generate novel and useful product. On the contrary, we define discipline-based work as the work demanded by the development disciplines and rules that are perceived to be useless to generate any novel things. And we use the time to do creative work (creative time) and the time to do discipline-based work (discipline-based time) to quantify creative work and discipline-based work in our empirical study.

Based on the educational projects used in the software architecture course, we designed an empirical study about creativity in software development. In order to increase the external validity and industrial relevance of this empirical study, we identified the following research requirements (RR) on educational projects:

RR1: Educational projects should provide an integrated software development process

RR2: Educational projects should simulate industrial context to improve the industrial relevance.

RR3: Educational projects should provide process to train students the necessary knowledge about programming skill and necessary concepts about software architecture to mitigate the differences of process experience between students and professionals before the beginning of the project in which research data are collected.

RR4: Educational projects should accept controlled process setting (independent variables). Concretely speaking, researchers plan to require half students to use UML-based documentation, and the other half not to use UML-based documentation.

RR5: Educational projects should provide multiple implementations on one set of requirements in order to do statistical results analysis.

Combining Pedagogical Requirements and Research Requirements to Design Educational Projects

Guided by the identified pedagogical requirements and research requirements, we designed two educational projects: a 3-week KWIC (KeyWord-In-Context) project and a 6-week Mobile Robot project. Both projects were selected from the standard example problems in software architecture community (Shaw et al., 1995) and were demanded to be implemented using Java. KWIC was a small project with the aim to let students be familiar with programming skills and software architecture concepts, and experienced the important and difficult issues of software architecture. While Mobile Robot was a big project, in which students lived through an integrated development process with simulated industrial context, from which research data were collected.

The 3-week KWIC Project

KWIC project was divided into two phases (Table 1): producing phase and evaluating phase. In producing phase, students were required to complete requirement specification, architecture design, and implementation (all tasks about detailed design, programming and testing were completed in implementation part). In architecture design part, students were demanded to give two architecture designs and select the better one to implement without being taught practical methods to accurately express and assess software architecture. In evaluating phase, students evaluated all the results produced in the first phase: requirement specification, architecture design and final implementation. We provided documentation templates to students in each phase.

Table1: The phase division in KWIC project and Mobile Robot project

Project	Phases setting				
KWIC	Producing				Evaluating
	Requirement specification	Architecture design	Implementation		
Mobile Robot	Requirement specification	Architecture design	Architecture assessment	Implementation	Postmortem analysis

KWIC project lasted three weeks. In the first two weeks, students completed the producing phase individually. And then they exchanged documents and final implementations to evaluate between every two or three students in the third week. We used another week to summarize the project and gave feedbacks to the students before the starting of Mobile Robot project.

In KWIC project, PR1, PR4 and RR3 are taken into account.

As to PR1, we postponed software architecture assessment work after the implementation phase with the aim to force students to experience much rework in implementation phase because they did not carry out software architecture assessment to find hidden defects in software architecture design. And in evaluating phase, we required students reporting their problems encountered in producing phase and asked them to assess whether these problems could be resolved through architecture assessment. Such process setting was used to convince students that they should explore and evaluate the process of software development to find hidden defects as early as possible, otherwise, if we take such defects to the later phase or stage, it will cost much more to repair or correct them, and also if they need to evaluate something, they must find a method to do it in time, otherwise the evaluating results can not improve anything. In addition, we did not teach any operable methods in this project to accurately express and assess software architecture in order to let students to understand the difficulty to describe and evaluate software architecture.

Regarding RR3, we expected students to master necessary knowledge and operations on software architecture through KWIC project. And we also expected to remind students of the knowledge of programming using Java. All of these were arranged with the aim to mitigate the differences of process experience between students and professionals.

In this project, we required all students developing KWIC project according to same requirements and evaluating other's implementation result with the aim to meet PR4 (our empirical research data did not come from KWIC project).

The 6-week Mobile Robot Project

Compared to KWIC project, Mobile Robot project was a more complex project which was divided into five phases: requirement specification, architecture design, architecture assessment, implementation (including detailed design, programming and testing) and post-mortem analysis (Table 1), and in each phase, we provided documentation template for students. We selected 'WSU Java Khepera Simulator ('WSU Java Khepera Simulator Home Page', 2003) as simulating environment for Khepera Robot ('Khepera Page', 2003).

In Mobile Robot project, students were divided into 12 groups randomly as developing unit, and exchanged their semi-finished products between every two sequential groups (Figure 1). Taking group 1 and group 2 as an example: after requirement specification phase, group 1 and group 2 exchanged their requirement specification documents and continued to complete architecture design task based on requirement specification documents they got. And the same actions happened after architecture design phase and architecture assessment phase. Finally after implementation

phase, the two groups carried out post-mortem analysis together. So each group attended every phase, while each of the 12 product lines was formed with each phase to be completed by one of the two sequential groups in turn.

Mobile Robot project lasted 6 weeks. Each phase used one week except for implementation phase which used two weeks.

We put the following three small questions into documentation template in each phase in order to collect research data: creative time, discipline-based time, and other time ('other time' was defined as time used to do the work which could not be classified as creative work or discipline-base work).

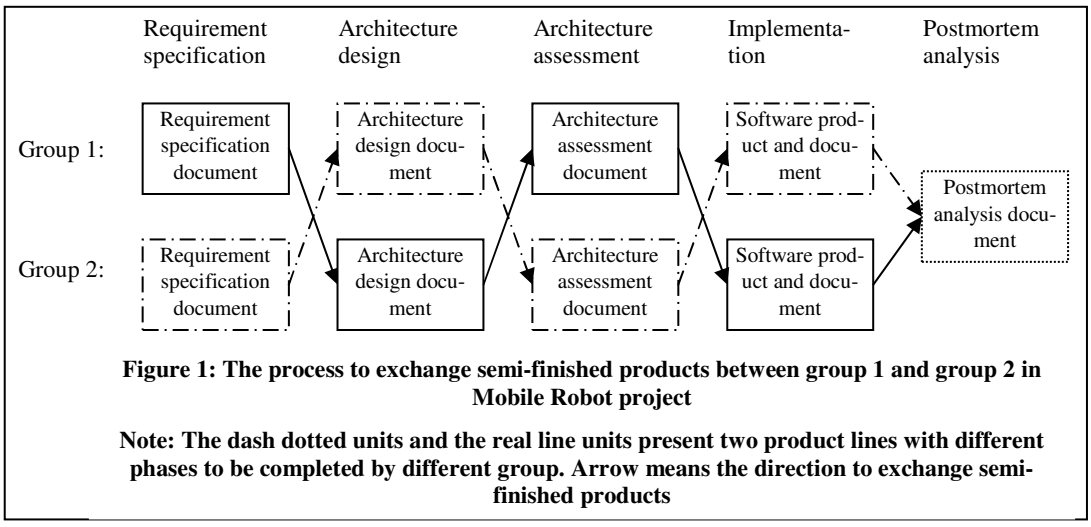

Figure 1: The process to exchange semi-finished products between group 1 and group 2 in Mobile Robot project

Note: The dash dotted units and the real line units present two product lines with different phases to be completed by different group. Arrow means the direction to exchange semi-finished products

This twin group setting and semi-finished product exchanging mechanism simulated the software development process in industrial context, in which all students experienced three roles involved in software development: customers, architects, system developers, and how they cooperated together to complete the a software development task: after students exchanged their requirement specification documents, each group (as developers) faced a requirement specification come from the other group (as customers) and made architecture design based on these requirements; after they exchanged architecture design documents, each group would check whether the architecture design made by other group met their defined specification (as customers) and whether the architecture design contained potential defects, how they should refine the architecture design to implement (as architects). And in implementation phase, each group got the evaluated and refined architecture design to implement (as developers). At last, the twin groups completed post-mortem analysis phase together to conclude and learn experiences from two product lines they took part in.

The integrated development process organization, multiple product lines setting, and simulated industrial context were designed to meet PR2, PR3, PR4, PR5, and RR1, RR2, RR5.

As to PR6 to teach students state-of-the-art techniques, we selected Philippe Kruchten's 4+1 Architectural Blueprints (Kruchten, 1995) and ATAM process (Kazman, Klein, & Clements, 1999) as the methods to express and access software architecture. Both these two methods are widely accepted in academic and industrial context. And currently, software development often concerns integrating some components developed by other companies or organizations (COTS-based development) (Shaw, 1999). In this project, we selected 'WSU Java Khepera Simulator' as the simulating environment for the Khepera Robot. This simulating environment can be seen as

COTS product, and students practiced software development process based on COTS products in this project to some degree.

In this project, not all pedagogical requirements and research requirements are consistent, so we have to trade off between conflicted requirements.

PR3, PR5 and RR1, RR2 required designing an integrated and simulated industrial context in which every development phase should get reasonable attention compared with industrial context. But PR2, PR7 demanded educational projects should concentrate on architecture description and assessment. So how much time should we allocate to architecture design phase and architecture assessment phase, and how much time should we give to other phases which relatively have less to do with software architecture education? To this point, PR3, PR5, RR1, and RR2 are conflict with PR2 and PR7. When trading off these two conflict parts, we were partial to PG2 and PR7, and put more attention to software architecture: dividing software architecture operation into two phases: architecture design and architecture assessment, each of them using one week; on the other hand, we combined the phases which had less to do with software architecture: detailed design, programming, testing into one phase named implementation, which used only two weeks.

Concerning RR3, in order to study whether there is distinct influence of UML-based documentation (used in requirement specification and architecture description) on students perception about how much time is devoted to creative work, we (researchers) had planed to require half groups to implement project using UML-based documentation and other half groups not using UML-based documentation. However, we selected Philippe Kruchten's 4+1 Architectural Blueprints as method to describe software architecture which is inclined to use UML-based diagrams to express software architecture. If we insisted on RR3 to ask half groups to use UML and other half groups not to use UML, PR7 will be harmed. The result of trading off RR3 and PR7 was to sacrifice research benefit (RR3) and allow all groups to decide themselves whether or not using UML-based documentation.

Also, we should consider that how much extra workload has put on students in order to execute empirical study. On one side, students' experience about research data collection during educational projects is consistent with PG6 because there are empirical studies and data collection processes in industrial context (Carver, Jaccheri, Morasca, & Shull, 2003). On the other side, too much workload about research data collection process may conflict with PG7. In Mobile Robot project, we put three simple questions to the documentation template, which added just a little workload.

Results Analysis

In the documentation templates of requirement specification, architecture design, architecture assessment and implementation, we added one part named 'encountered problems' in order to collect data about the problems students met during each phase of these two projects (In producing phase in KWIC project, we added 'encountered problems' to each part of the documentation template: requirement specification, architecture, implementation.)

In the documentation template of evaluating phase in KWIC project and that of postmortem analysis phase in Mobile Robot project, we put some questions, such as, 'what went right' and 'what went wrong' in order to extract their comments about what students have learned and experienced.

In this section, we will use students' encountered questions and comments as evidence to illustrate whether our educational projects meeting the pedagogical requirements and research requirements.

KWIC Project

During KWIC project, students put forward many questions (Q) and complains, which can be formalized into following five categories:

Q1: 'How can I describe software architecture accurately since architecture phase is in a so early stage?'

Q2: 'What are the relationships between view, viewpoint and scenario concerned with software architecture?'

Q3: 'Since I am required to evaluate two architecture designs and select one from them, is it meaningful to do this through thinking and discussing, and is there any quantitative or practical methods to evaluate them?'

Q4: 'Though it is a small project, but it cost too much time to complete it because I have to spend quite a lot time to read materials about software architecture and to be familiar with the knowledge of programming using Java.'

Q5: 'Mr. (Miss.) … has the different understanding to view, viewpoint or scenario. As to me, view, viewpoint, or scenario means…'

The first three questions show us that students had begun trying to grasp the concepts involved in software architecture (for example, the concepts of viewpoint, view, and scenario). And they had found the important and difficult issues involved in software architecture: how to describe software architecture accurately and how to evaluate software architecture efficiently. With the experience to these questions, students would have much intention to study operable methods to resolve them, which was the main content in Mobile Robot project.

We had also expected students to experience many conflicts between architecture design and system implementation, and live through much reworking in implementation phase in KWIC project because they did not do software architecture assessment to correct defects and conflicts hidden in software architecture design. However, we could not find any direct evidence to support they had experienced such conflicts and reworking. One possible reason is that reworking is blurred by the programming problems shown in Q4. Another reason can be that KWIC project is so small (400-500 lines) that hidden defects and conflicts can be found without software architecture assessment. We believe that PR1 will be satisfied better if we provided students with a larger project, or we provided them with a completed architecture design of a large or complex project with several defects or conflicts hidden inside (without informing them) and forced them to live through the implementation phase to experience the reworking or conflict.

Q4 shows us that students had used quite a lot time to study the knowledge about software architecture and to be familiar with the skills to program using Java which mitigates the process experience gap between students and professional, so RR3 was satisfied through this project. And Q5 tells us KWIC project satisfied PR4.

Mobile Robot Project

In Mobile Robot project, students brought forward fewer complains or questions than that in KWIC project, and all 12 groups gave the positive evaluation of Mobile Robot project in post-mortem analysis documents and thought this project could help them mastering the knowledge and methods about software architecture. Here, we will cite some students' comments (C) as evidence to illustrate whether our pedagogical requirements and research requirements are met (the satisfied requirements are listed in the bracket following each comment):

C1: 'We got a good understanding of what viewpoints and views are.' (PR2)

C2: ‘The project has been a good way to get an alternative understanding of the syllabus. There are many aspects of software engineering you need to experience to understand – you can’t just learn it by reading a book.’ (PR2, PR3)

C3: ‘We have learned better techniques to evaluate software architecture. We feel our newly acquired knowledge and experience about software architecture will be useful in larger projects.’ (PR2, PR3)

C4: ‘We have gained an understanding of the importance of software architecture through developing a project like this one. Also, it is clearer now that the implementation stage is far easier with a solid architectural fundament.’ (PR2, PR3)

C5: ‘We are actually quite satisfied with how the implementation phase turned out. Our choice of architecture made it easy to implement the required functionality.’ (PR2, PR3)

C6: ‘The most successful part of our ATAM adoption was the scenario, analysis question, and consequence phases. These phases are proved to be a useful tool for discovering both advantages and weaknesses of the architecture.’ (PR2, PR6)

C7: ‘The presentations from each group were a good way to get insight into how other groups had solved the problem they formulated.’ (PR4)

C8: ‘The project has also given us more experience in working in a group and solving problems as a group.’ (PR5, RR2)

C9: ‘The exchange of documents between groups was a clever trick to simulate a larger project than what our limited time and resources allowed in this course, and helped to discover missing or implicit information. If you are to test your own requirements, you might not notice if they are not properly specified.’ (PR3, PR4, PR5, RR2)

C10: ‘Getting to know the limitations and opportunities of the COTS even better…’ (PR6)

There is no complain about they had spent too much time and energy to complete research tasks, which is an evidence to support that our projects design meets PR7.

In this project, we collected our empirical research data from multiple implementation instances based on the same set of requirement specifications (RR5) in an integrated (RR1) and simulated industrial context (RR2) project with the students who had improved their knowledge level about software architecture and programming skills (RR3) in KWIC project.

The empirical research result about creativity in software development takes the form of four hypotheses (H).

H1: In software development, there is most creative work in implementation phase and least creative work in postmortem analysis phase.

H2: UML-based documentation can promote students to do more creative work in requirement specification phase and architecture design phase.

H3: More creative work can not accelerate or decelerate development speed.

H4: Students prefer the phases which include more creative work.

You can get more detailed information about this empirical study in Gu and Tong (2004).

Conclusions

In this paper, we provided a successful case to resolve the challenge that ‘how should we design educational projects successfully from both educational perspective and research perspective’.

First, based on the literature study, we identified the pedagogical requirements on educational projects from the perspective of software architecture education, and the research requirements from the perspective of empirical study. And then, two projects, KWIC and Mobile Robot, were designed to meet these two types of requirements. Students' questions and comments were used to evaluate the results of our design. As summarized in Table 2, only one of the twelve requirements was not met, and eight requirements were satisfied fully.

Though our identified pedagogical requirements and research requirements are associated with software architecture education and empirical study about creativity in software development, we believe these requirements and our methods to satisfy them have reference benefits to other software engineering courses in which an empirical study is scheduled. We also plan to use these methods to design educational projects in the next year's software architecture course in which some empirical studies will be carried out.

Table 2: The results about whether each requirement is satisfied

Requirements	Methods	Result
PR1: Teach students the importance and the difficult topics of software architecture at the beginning of this course	In KWIC project, students were asked to complete this project without teaching them the operable methods about how to describe and assess software architecture, and we designedly postpone architecture assessment after implementation in order to let them go through more reworking and conflicts during implementation.	Partially Satisfied
PR2: Give students the chance to experience how to describe software architecture accurately, and how to assess software architecture effectively.	In Mobile Robot project, students completed software architecture description and assessment using operable methods.	Satisfied
PR3: Provide students a chance to experience how software architecture is integrated into the whole software development.	In Mobile Robot project, software architecture description and assessment were integrated into a whole software development process.	Partially Satisfied
PR4: Let students provide multiple software architecture designs, architecture assessments and system implementations on one project so they can compare and compete to each other.	In both KWIC and Mobile Robot project, students completed each project individually or in group, which provided multiple implementations to each project.	Satisfied
PR5: Provide an industrial context or a simulated industrial context.	In Mobile Robot project, twin group setting and semi-finished product exchanging mechanisms were used to simulate industrial context.	Satisfied

PR6: Use the state-of-the-art techniques used in industrial context currently.	In Mobile Robot project, we used Philippe Kruchten's 4+1 Architectural Blueprints and ATAM process to express and access software architecture. And we asked students to implement mobile robot project based 'WSU Java Khepera Simulator' to experience the COTS based software development.	Satisfied
PR7: Not use too much time on the topics which have little to do with software architecture issues.	Simple data collection methods; Integrating detailed design, programming, testing into one phase named implementation, which used only two weeks	Satisfied
RR1: Should provide an integrated software development process	Providing a integrated software development process though more attention is put on software architecture	Partially Satisfied
RR2: Should simulate industrial context to improve the industrial relevance.	Same with the methods to meet PR5	Satisfied
RR3: Should provide process to train students the necessary knowledge and skills before the beginning of the project in which research data are collected	In KWIC, students learned knowledge about software architecture and reminded themselves of programming using Java	Satisfied
RR4: Require half students to use UML-based documentation, and the other half not to use UML-based documentation.	This requirement was sacrificed because the adoption of Philippe Kruchten's 4+1 Architectural Blueprints provided a convenient way to use UML-based documentation to describe their architecture design	Not satisfied
RR5: Should provide multiple implementations on one set of requirements in order to do statistical analysis.	Same with the methods to meet PR4	Satisfied

References

Andersen, R., Conradi, R., Krogstie, J., Sindre, G., & Sølvberg, A. (1994). Project courses at the NTH: 20 years of experience. *Conference on Software Engineering Education*, San Antonio, Texas, USA.

Bernstein, L., & Klappholz, D. (2001). Teaching old software dogs, old tricks. *Software Engineering Notes, 26* (2), 33-34.

Carver, J., Jaccheri, L., Morasca, S., & Shull, F. (2003). Using empirical studies during software courses. *Empirical Methods and Studies in Software Engineering - experiences from ESERNET* (pp. 81-103). Springer.

Carver, J., Shull, F., & Basili, V. (2003). Observational studies to accelerate process experience in classroom studies: An evaluation. *International Symposium on Empirical Software Engineering*, Rome.

Cooper, D.R., & Schindler, P.S. (2003). Ethics in business research. *Business Research Methods* (pp. 118-141). New York: McGraw-Hill/Irwin.

Dybå, T. (2000). Improvisation in small software organizations. *IEEE Software, 17* (5), 82 - 87.

Glass, R.L. (1995). *Software creativity*. Prentice Hall.

Gu, M., & Tong, X. (2004). Towards hypotheses on creativity in software development. *5th Int'l Conf. on Product Focused Software Process Improvement* (PROFES'2004), 5-8 April, 2004.

Humphrey, W.S. (1987). *Managing for innovation: Leading technical people*. Prentice Hall Trade.

Kazman, R., Klein, M., & Clements, P. (1999). Evaluating software architectures for real-time systems. *Annals of Software Engineering*.

K. Team, (2003) Khepera Page. Retrieved from http://www.k-team.com

Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software, 12* (6). 42 - 52.

Lethbridge, T. C. (1998). A survey of the relevance of computer science and software engineering education. *Conference on Software Engineering Education and Training*, Atlanta, GA.

Royce, W., Boehm, B., & Druffel, C. (1994). Employing UNAS technology for software architecture education at the University of Southern California. *The Eleventh Annual Washington Ada Symposium & Summer ACM Sigada Meeting on Ada*, McLean, Virginia, United States.

Schneider, J., & Johnston, L. (2003). eXtreme programming at universities - An educational perspective. *25th International Conference on Software Engineering*, Portland, Oregon.

Shaw, M. (1999). Architectural requirements computing with coalitions of resources. Position paper for *First Working IFIP Conference on Software Architecture*. 2003.

Shaw, M. (2000). Software engineering education: A roadmap. In A. Finkelstein, *The future of software engineering* (pp. 371-380). New York, NY: ACM Press.

Shaw, M., Garlan, D., Allen, R., Klein, D., Ockerbloom, J., Scott, C., et al. (1995). candidate model problems in software architecture. Discussion draft 1.3 in *Circulation for Development of Community Consensus*.

Winograd, T. (1996). *Bring design to software*. Addison Wesley.

Wright State University (2003), WSU Java Khepera Simulator Home Page. Retreived from http://gozer.cs.wright.edu/classes/ceg499/sim/sim.html

Biography

Mingyang Gu is a Ph.D. student of the artificial intelligence and learning group at the department of computer and information science, Norwegian University of Science and Technology (NTNU), Norway. He received his B.E. and M.E. degrees respectively in 1999 and 2002 from RenMin University of China. Currently, his research interests include creativity and constraints in software development, Conversational Case-Based Reasoning and knowledge-intensive Case-Based Reasoning.