Planning an Empirical Experiment To Evaluate The Effects Of Pair Work On The Design Phase Of The Software Lifecycle

Hiyam Al-Kilidar[1], Ross Jeffery[1], Aybuke Aurum[2], Cat Kutay[1]
[1]School of Computer Science and Engineering
[2]School of Information Systems, Technology and Management
University of New South Wales
Sydney, NSW, 2052 Australia
{hiyama | rossj | ckutay@cse.unsw.edu.au}, aybuke@unsw.edu.au

Abstract

This report presents the details of an empirical experiment designed to evaluate the effects of Pair Work on the design phase of software development lifecycle. The experiment is designed to investigate the effects of pair work on the quality of design products and whether the pair work approach in the design process is more efficient or cost effective than individual work approach. The aims of the experiment are to compare the quality of the design products produced by pair designers and individual designers as well as compare the efficiency and cost effectiveness of the pair work approach and the individual work approaches in the design process. In addition, the experiment studies the partner's expectations and practices during the pair work experience. The experimental hypotheses, design, inputs, outputs, and evaluation measures will be described.

Introduction

The design and development process in software lifecycle involves the explanation of how design products meet the requirements. Software design products represent different views of a software system under development such as architectural, logical, structural etc. Those views are used as communication tools between stakeholders to understand, communicate, manage and maintain software systems. High level design representations provide a common conception of a system structure and include abstractions of its components and the relationships between them. It also allows for the manifestation of early decisions about prioritizing concerns of the system qualities. Lower levels (detailed) design representations illustrate the program specifications that are later passed to coding and implementation phases including details of the internal specifications and behaviours of the system components and modules, data definitions, inter-component and/or module interface documentation etc. The larger the system the more crucial are the design phase products for the analysis, communication, management and maintenance purposes.[1, 2]

The products of the design process form the plans for the coding phase performed by programmers [3]. To the best of our knowledge, there are no studies to evaluate the effects of pair work approach on the process and products of the design phase of the software development lifecycle. However, literature revealed that the product and process of pair programming work in the software development lifecycle have been examined in experiments such as [4-8]. Those experiments examined the effects of pair programming compared to individual programming on the quality of produced code, as well as investigated some aspects of the effects of the pair work process and its effects on the partners' experiences. These experiments only examined the quality of the code and the coding process and did not investigate the products and process of the design phase that precedes the coding phase.

Software Design Phase

Design is an activity that is concerned with producing representations of programs that reflect requirements. The Design phase is commonly divided into stages that describe the system at different levels of abstraction to translate the user requirements into source code. The development process is characterized by four distinct phases of Requirement Analysis, Design, Coding and Testing. These phases can be sequential or iterative and each phase is mainly charcterised by both its process and product. The software design phase aim to illustrate, control and manage the massive amount of detail and complexities (relationships of many sorts between many parts) involved in the creation of systems [9].

Design Process

The design process transforms an input document of user requirement specification into design documents output with specifications of algorithms and data structures that form the basis for the coding phase[10-12].
An initial design process begins with activities such as the study, understanding and analysis of the problem from different angles by specifying and analyzing the users requirements and hence, the identification of alternate preliminary design strategies and software architectures. This is followed by a detailed design stage which refines the architectural representations and deciding on a design methodology /practice. The third design stage will detail the procedural design and data structure and specifications of algorithms and development of model. These stages often overlap and feedback is used to refine earlier outputs making the design process a highly iterative one. [10, 11, 13, 14]

Design Products

The representations of the design process can vary between high-level details of an overall diagram representation of control-flow to very near to actual programs such as a metacode or pseudo code description of a program. Achieving such an outcome evolves around making major decisions of issues such as preparation of abstracts of operations, system data, interface and control linkages between parts of system and other systems, choice between design alternatives, trade offs to meet requirements based on conditions and constraints etc. [9].

The Human factor in Software Engineering

The software Engineering process for developing software systems involve the participation of people as system users, specifiers, analysts, designers, programmers, managers, technicians etc. Limitations and abilities of people should be taken into account when designing a system for it to be used in the best possible way. In addition, an understanding of human interaction patterns can help identify possible ways of increasing productivity and effectiveness of processes [11].

Individual and collaborative Work

The traditional popular practice of executing the development stages is through teams of professionals. The teams' size can range between 2 and several hundred people. The mechanism of executing the assigned tasks in a team situation, usually involve partitioning the work and assigning team members to carry out different tasks as individuals and then communicate/ combine their outputs to/ with other members of the team [11, 15].

Pair Work

Pair work is a form of collaborative or group work that involves two people working together as one unit to produce a single output. The feature that differentiates pair work from other collaborative or group work is that instead of partitioning the work into different tasks where each group member performs a different task, in pair work both partners actually perform each task together, making it possible to create and continuously review what is being created. It is claimed that such a process makes it possible to identify and fix defects early, hence, produce a better quality product [4, 5, 7, 16, 17]. It is also suggested that pair work enhances interactions and communication channels between partners, provides opportunities for learning, as well as making the whole process more enjoyable than working individually [6, 17, 18].

Pair Programming

The concept of programming in pairs was known since the 1950's [16], but has recently been "re-invented" by extreme programming. In extreme programming of agile methods, code is produced as a result of collaboration between two programmers. The pair programming technique is identified as two programmers working side by side on the same problem using one computer to produce *one* artifact (algorithm, code). One partner acts as a driver who controls the creation tool (e.g. pencil, mouse, keyboard) and writes code, while the second partner reviews the drivers' work and acts as an observer and/or quality assuror who continuously and actively examines the driver's work, watching for defects, thinking of alternatives, looking up resources, and considering strategic implications. Partners deliberately switch roles periodically, and both are equal active participants in the process at all times and both claim equal responsibility for their product [4, 6, 8, 16, 17, 19-21].

Some experiments such as [4, 6, 7] suggest that benefits of pair programming include the early detection of many mistakes with a statistically lower end defect content and shorter code length. In addition, people involved in pair work learn significantly more about the system development

and work and talk together more often, gaining better team dynamics and information flow as well as enjoy their work more and solve problems earlier and faster. The project ends up with more people understanding each part of the system [4, 6, 7, 16, 17, 21, 22].

Related Experiments

Literature review of pair work experiments such as [4, 6, 7] have shown that pair programming in software development processes yields better products in less development time and happier, more confidant programmers who outperformed individual programmers. The same experiments have also shown that pair programming also contributed to factors such as improvement in software quality and reduction in time to market, which are factors of the success of XP[4]. It is also suggested that pair programming reduces maintenance expenses and improve customer satisfaction. Other experiments such as [8] has shown that pair programming leads to more stable solutions, however the same experiment has also shown that pair programming is a rather expensive technology contrasting the results of [7] that pair programmers needed less time to accomplish their tasks.

Pair Work in the Design Phase

This research provides an extension to the previous studies of pair programming. Although it is suggested in literature [4, 6, 16, 17, 21, 22] that the pair programming technique is not only applicable to the coding phase of the development lifecycle, but also include the design phase, no experiment has targeted the design phase process and evaluated the design outcome as a separate product component from the code. What is usually evaluated and compared in experiments are code lengths and defects per line of code as measures of the developed software quality [4, 6-8].

Design is an activity that involves making decisions based on scientific principles, technical information and imagination about the creation and organization of solution outlines (architecture, logical, structural etc) for a problem. [3, 9, 23]. The design process is concerned with describing how design product meets the requirement, while the design products are used by different stakeholders in the software system as tools of analysis, management, communication and maintenance as well as provide the plans to the coding phase by programmers [1, 3]. Therefore, the design phase outputs vary in their level of abstraction from conceptually higher level of abstraction to detailed lower levels of system representations such as pseudo codes or structured English.

Personal preferences and styles in making decisions are more apparent in design activities than other activities such as coding ones. More than one acceptable solution to any problem is quite common and therefore, conducting experiments to compare between the design processes and the quality of the design products are more difficult than conducting similar experiments to compare between programming/ code processes and quality of products.

The difficulties associated with the conduct of design experiments include, but are not limited to, the degree of control the researcher has over the design activities compared to experiments of coding activities. This implies that some design elements may not be clear enough to be communicated or documented during the design process. In the case of designers working in pairs, an effort might be necessary to work on and further develop communication and negotiation skills to explain and exchange ideas and resolve differences of opinion between partners may prove to be necessary.

Other difficulties are associated with the evaluation of the quality of design products. Unlike the quality of code that can be objectively determined by the number of defects per line of code, the quality of design representations/products is more complex. Design representations vary in form and characteristics to represent different views of the system [3]. Evaluating the quality of representations is a highly subjective issue since designs are, by nature, subjected to personal

preferences and style of designers. In order to reduce subjectivity, the quality of the design products will be based on ISO/IEC 9126 Software Engineering - Product Quality, Internal Quality Characteristics and Metrics [24, 25].

Quality and Quality Measures of Software Design

Quality is a term that is hard to define precisely. Many definitions have been introduced in the literature to define quality [26]. The International Standards Organization (ISO) formally defines Quality as "The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs". A definition of software quality proposed by Pressman is "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software"[13]. Since most of the quality factors such as Correctness, Reliability, Usability, etc. are highly subjective, it follows that the quality measures are sometimes impossible to develop let alone objectively measure. The International Organization for Standardization ISO and the International Electro-technical Commission IEC have produced the ISO/IEC9126 International Software Engineering –Product Quality Standards that is divided into four parts. Part1 (ISO/ IEC 9126-1) is the Quality Model, Part2 (ISO/ IEC 9126-2) External Metrics, Part3 (ISO/ IEC 9126-3) Internal Metrics, and Part4 (ISO/ IEC 9126-4) for Quality in Use Metrics[24, 25, 27, 28].

Please refer to Fig. 1 for the quality model as described in ISO/IEC 9126 [25]. Each one of the six quality characteristics in Fig 1 is further described by a set of sub charectistics. Each one of these subcharacteristics is defined in terms of external and internal metrics [24, 27]. For each metric, the purpose, method of application, measurement, formula and data element computations, Interpretation of measured value, metric scale type, measure type, input to measurement, reference and target audience is identified and described.

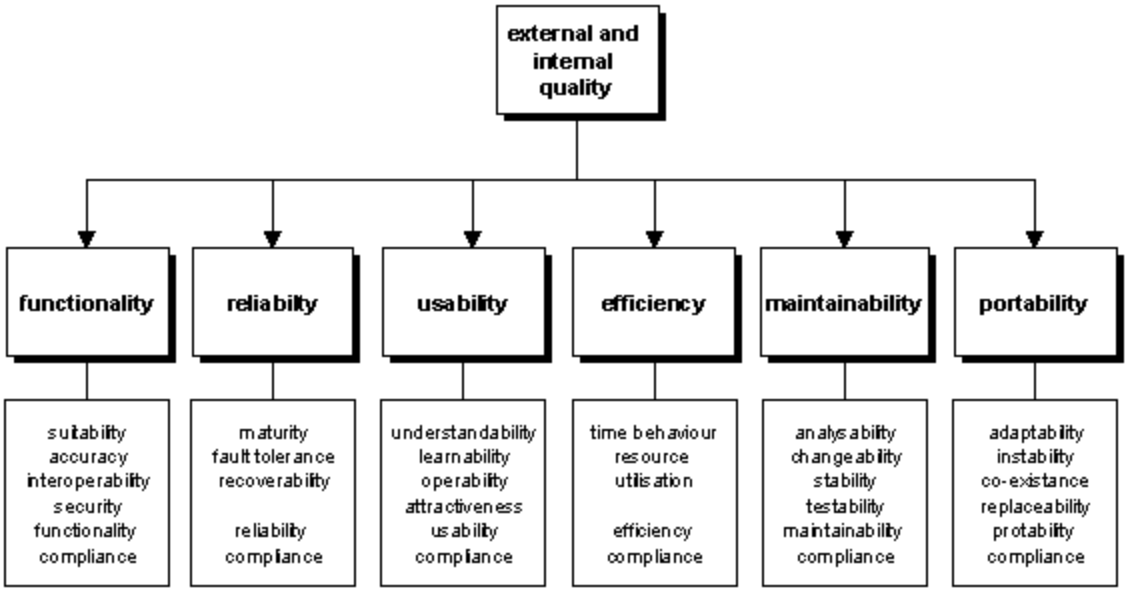

Fig 1: Quality Model for external and internal quality

For the purposes of evaluating the quality of design products, the ISO/IEC 9126 standards of Internal Metrics [24] identify metrics of subcharacteristics of the Functionality, Usability and Portability characteristics to be gathered from the design as the source of input to the measurement. Therefore, in evaluating the design products of our proposed experiment, these three quality characteristics will be evaluated.

The *Functionality* sub-characteristics to be evaluated in this experiment are *suitability* which is measured by the metrics of *functional adequacy*, and *functional implementation coverage*, *security* which is measured by metrics of *access auditability* and *access controllability* and *functionality compliance* that checks for satisfaction of the DFD rules. For the *Usability* sub-characteristics to be evaluated the *understandability* measured by *completeness of description*, *evident functions*, and *function understandability* metrics are used, in addition to the *usability compliance* metric. For the *Portability* characteristic, the metrics of *hardware environmental adaptability* and *system software environmental adaptability* are measured.

Details of the Experiment designed to evaluate the process and product of design phase

An experiment is planned to evaluate the design process and products of individual and pair designers. This experiment introduces the concept of pair work in the design phase to mirror the pair programming concept. Designing in pairs is the representation of two designers jointly working on the same desk/computer to produce one set of design documents. Both designers are responsible for the creation and modification made to the design deliverables. One designer acts as a driver who composes the specifications or design ideas and representations, while the second designer continuously and actively observes, reviews and guides the driver's work. The two designers deliberately switch roles periodically, and both are equal active participants throughout the design process at all times. Following the strategy of XP in writing test before code [29], the design products are to be tested and evaluated according to the internal quality measures identified in ISO/IEC 9126-3 [24]. Metrics of *functionality, usability* and *portability* characteristics as identified in [24, 25] will be measured for the design products since the ISO/IEC 9126-3 [24] standards identify the design phase to be the source of input to the measurement of metrics of these quality characteristics.

Aims of experiment

1. Compare between the quality of the design products produced by pair designers and individual designers
2. Compare between pair work and individual work approaches in the design process.
3. Gain knowledge about the partners' expectations and practices during the pair work experience.

Research Questions and Hypotheses

This section presents the Research Questions (RQ) and the Hypotheses (H) to be tested in the experiment, and the types of questions asked to illicit the relevant information.

RQ1. What effects does pair work have on the quality of designs?

The quality characteristics to be investigated for the design products are *Functionality*, *Usability* and *Portability* based on the ISO/IEC 9126-1 [25]and 9126-3 [24] standards. Therefore, the RQ1 was divided into three hypotheses:

H1. Pair work approach produces a more functional Design-Product than individual work approach.

RQ1a0 If H1 is false, what factors contribute to pair work being no better than individual work approach in producing a more functional design product?

RQ1a If H1 is true, what factors contribute to pair work being better than individual work approach in producing a more functional design product?

H2. Pair work approach produces a more usable Design-Product than individual work approach.

RQ1b0 If H2 is false, why is pair work no better than individual work approach in producing a more usable design product?

RQ1b If H2 if true, why is pair work better than individual work approach in producing a more usable design product?

H3. Pair work approach produces a more portable Design-Product than individual work approach.

RQ1c0 If H3 is false, why is pair work no better than individual work approach in producing a more portable design product?

RQ1c If H3 is true, why is pair work better than individual work approach in producing a more portable design product?

RQ2. What effect does pair work have on the efficiency and cost effectiveness of the design process?

The design process is to be investigated for the efficiency and cost effectiveness according to the two following hypotheses:

H4. The pair work approach is more time efficient than individual work approach to produce software designs.

RQ2a0 If H4 is false, why is the pair work approach no more time efficient than individual work approach to produce software designs?

RQ2a If H4 is true, why is pair work approach more time efficient than individual work approach to produce software designs?

H5. The pair work approach is more cost effective than individual work approach in producing software designs.

RQ2b0 If H5 is false, why is the pair work approach no more cost effective than individual work approach in producing software designs?

RQ2b If H5 is true, why is pair work approach more cost effective than individual work approach in producing software designs?

RQ3. What factors affect the experience and practices of partners in pair work?

The process of pair work is influenced by many factors. Hypotheses representing such factors are formulated to be analysed from questions to be given to partners after the completion of the pair

work process asking about their pair work experience and how it may differ from individual work, and what historical factors these differences may depend on. The hypotheses state that each of the following affects pair performance:
H6. The extent to which partners followed the pair work process as compared to other types of collaborative work.
H7. Whether the partners had previously worked with the same partner.
H8. The way differences in opinion were dealt with.
H9. The emotive affects of pair work.
H10. The learning experience of pair work as perceived by the partners.
H11. Differences in levels of competence.
H12. The gender combinations of pairs.

Experiment Scope

The experiment will run as part of the course work for Software Project Management Class in the School of Computer Science and Engineering at the University of New South Wales. The students in the course will be asked to produce design products as well as plan and track their design activities using Microsoft Project. All subjects will be asked to produce the same design deliverables.

Experiment population

The subjects of the experiment are approximately 225 undergraduate computer engineering students in their second semester of 3rd year. All subjects will have passed at least one design course. The majority of them are trained in design methods of Data Flow Diagrams (DFD) and Entity Relationship Diagrams (ERD). For those with no prior DFD knowledge, tutorials on DFD are arranged for them in the first week of the semester before the start of experiment. All subjects will also be given instructions on pair work and will be asked to read the article by Williams and Kessler [17]. The subjects will also be given a quiz on pair work practices prior to the beginning of experiment to ensure they put the effort into learning the pair work practices.

Due to the relatively short period of time allocated to produce each deliverable, subjects will be given the option to choose their own partners to reduce personality clash issues between partners. This is based on the assumption that subjects will pair with colleagues that they have no personal problems working with.

Information about subjects' weighted average marks (WAMs), as calculated by the University, and their gender will be collected to form part of the analyses of results. The assumptions are that WAMs are a competence measurement of performance and subjects with higher WAMs are better designers than subjects with lower WAMs. This allows the subjects to be classified into Good Designers (GD), Average Designers (AD) and Novice Designers (ND). The classification will be used for analysis of pair performance based on the combination of designer levels (GD-GD, GD-AD, GD-ND, AD-AD, AD-ND, ND-ND). In addition, it is assumed that gender issues have no effect on performance. Pairs of different gender combinations (male/male, male/female, or female/female) can be analysed for practices and communication issues within the pairs.

Experiment Context

The subjects will be asked to design part of a realistic network project management tool that is capable of effectively and efficiently manage a large number of autonomous collaborating partners. The part to be designed by the subjects include two modules, a planning module and a tracking one. The vision is of a system in which there is remote management of separate production units coupled with expert systems that monitor the complete production process and are able to take corrective, perfective and adaptive action, as required. The principle technology enablers for this system are:

- The Internet, which will provide communication between remote users and centralized server.
- The expert system that enables the skills of expert producers to be made available to many other producers.

The subjects will be given detailed information about the system context and user requirements together with a set of conceptual data models in ERD format. The ERD's are used to identify principal entities and the principal relationships between those entities that must be managed in the database, along with the values of the attributes of those entities and relationships.

Also given to the students will be a set of use cases that describe the system's behaviour to be used as functional and design scope [30] for both the planning and the tracking modules to be designed. The use cases will provide enough information to show system behaviour without describing system interface design.

Pilot Study

A pilot study will run using the tutors as test cases. This will take place prior to the start of the experiment. The objective of the pilot study is to verify the feasibility of the experiment in terms of the difficulty of the design and the time required to complete a reasonable design from the given requirements. It also enables the researchers to develop a set of evaluation metrics for the design that are suitable for the requirements and design format used and at the same time comply with ISO/IEC 9126 standards.

Course of the Experiment

The experiment itself will run over a period of 8 weeks. Subjects will be randomly divided into 14 tutorial sessions. Each tutorial session will consist of 12-18 subjects. In order to give all subjects the opportunity to practice working in pairs as well as individually, the experiment task is divided into designing two modules, the planning module and the tracking module. For the planning module, two use cases are given and for the tracking module, three use cases are given.

Each student will work as an individual and as a partner in a pair on one of the design modules. Subjects who will design the planning module individually will be required to pair with another individual when designing the tracking module. Subjects who work in pairs designing the planning module will be asked to design the tracking module individually.

Following the strategy of writing test before code of XP [29], the first design activity the subjects will be asked to consider is to establish the quality plan for their design deliverables. The subjects will be given quality metrics derived from ISO/IEC quality standards [24, 25] explaining what is to be achieved, and they will be asked to establish how each metric is it to be measured. The subjects will be required to consider these points for how their design is to be evaluated, and to show which part of their design deliverables is representative of each quality metric.

In designing the planning module, subjects will be given 3 weeks to complete their task. Table 1 illustrates the allocation and numbers of participants in the experiment for producing the planning module.

Group A (Control)	75 Individual Designers
Group B (Treatment)	75 Pairs of Designers

Table1: 2x1 Experimental design for Planning Module

For the tracking module, the tutorial groups will be switched. Those who worked individually will be asked to work in pairs, and vice-versa. This exercise will last for 3 weeks and subjects are

asked to produce the same design documents as the planning module. Table 2 illustrates the allocation and numbers of participants in the experiment for producing the tracking module.

Group A (Control)	150 Individual Designers
Group B (Treatment)	75 Pair of Designers

Table2: 2x1 experimental design for Tracking Module

Data Collection and Evaluation

The design outputs of the planning and tracking modules will include a process model in DFD format, a database storage model in table format, an interface model with screen dumps of system, as well as a time log of all their activities using Microsoft Project Reports.

The Data Flow Diagrams (DFD's), Data Tables and Interface Designs are design representations of both the planning and the tracking modules. The DFD's provide a network representation of the structure and logic of the system partitioning it into its components according to the use cases. The subjects will be asked to start with a context level DFD, then level 0, then detail and expand the processes until 3rd level according to the use cases, and provide Structure English or pseudo code for the 3rd level diagram. In the data table deliverable, the subjects are asked to specify the attributes of the entities that they use for each entity. Screen shots of system functionality (behaviour) represent the interface deliverable.

Subjects are asked to electronically submit all their design deliverables together with time logs of all their design activities. Partners in pairs will be asked to have just one submission and they would each get the same mark. In order to eliminate bias in evaluation of designs, the design deliverables will be double blindly assessed to ensure that the assessor does not know if an individual or a pair produced the designs.

For Hypotheses 1-3

Hypotheses H1-H3 test for the quality of the design product from the developer's point of view. Evaluation of quality characteristics is usually a subjective issue. In this experiment an effort is made to reduce subjectivity of quality assessment of design products by the use of ISO/IEC 9126 set of Software engineering - Product quality standards [24, 25]. The standards have developed metrics that can be measured by tallying the number items represented in the design documents. This feature gives more credibility towards reducing subjectivity in evaluating designs. Based on these ISO/IEC 9126 Software Engineering –Product Quality standards [24, 25], the internal quality characteristics [24]are evaluated for this experiment. The experiment is restricted to the design phase of the software life cycle, therefore, Internal quality characteristics and sub-characteristics of the design products will be tested for *functionality, usability* and *portability* since according to ISO/IEC 9126- 3 of internal metrics [24]. These three quality characteristics and their subcharacteristics as shown in Fig 2 and described earlier are the ones associated with the design phase.

Tables 3 and 4 describe the guidelines for evaluating the planning and the tracking modules. The quality attributes to be verified were based on the metrics in [24], and the scope of measure describing the source and tally of the measures.

			Scope of Measure			Design Product Measured		
The attribute of quality to be verified	**Goal : What to achieve in this area**	**Metric : How to measure this**	**Use Case 2**	**Use Case 3**	**Overall**	**DFD**	**Interface**	**Data Table**
Functional Metrics								
Functional Adequacy	Whether all functionality exists	Interface functionality deals with forms/screens/reports/menus etc 2.1 Risk Rating screen occurs 24 hours after Risk entry 2.2 Contingency plan request only if risk>2 3.1 Rating Evaluation is time limited	X : 2	X : 1			X	
		DFD should have processes for each data manipulation 2.1 Save new risk for review 2.2 Receive feedback and link to risk for review 2.3 If risk> 2 add to permanent risk list 2.4 Save Contingency Plan 2.5 Save Likelihood and Impact 2.6 Eliminate Risk<2 3.1 Save and add risk evaluations	X : 6	X : 1		X		
Total Score of Functional Adequacy Metric = Actual DFD/7+ Actual Interface/3								
Functional Implementation Coverage	Whether the functions are enough to do what the client wants	Interface: should show the screens/links.	X : 6	X : 1			X	
		2.1 Read Existing Risk List 2.2 Enter New Risk 2.3 Read Risk Rating Feedback						

		2.4 Enter Contingency Plan 2.5 Enter Risk Likelihood 2.6 Enter Risk Impact 3.1 Enter Risk Rating						
		DFD has flows for all data reports 2.1 Report existing Risks from data base 2.2 Report Risk Rating from data base 2.3 Report Contingency plan (likelihood and impact) to Management 3.1 Send Risk Evaluation Request 3.2 Save Risk Evaluation value in data base.	X : 3	X : 2		X		
		Data Table should have data attributes in data base 2.1 Risk evaluation should be in data table 3.1 Risk time of entry should be in data table	X: 1	X: 1				X
Total Score of Functional Implementation Coverage Metric= Actual DataTable/2 + Actual DFD/5+ Actual Interface/7								
Access auditability	Records should be kept of participants contributions.	Measure: Exists: Login screen 1/3 Login process 1/3 User data attached to database data 1/3			X	 X : 1/3	X : 1/3	 X : 1/3
Access controllability	Access to system should be controlled to some data.	Measure: Exists Data link between login process and all report data for each screen in Use Case	X	X		X		X
Functional compliance	DFD rules	Data Flows are maintained between levels			X	X		
		Data base consistency : only one process writes to each data base				X		
Usability Metrics								

Completeness of description	All functions described in an understandable manner	Measure: Step through Use Case and subtract one point for ambiguity as to data presented in screen at any point	X : total is 6	X : total is 1			X	
Evident Functions	User able to understand how to perform tasks from design	Measure: Step through Use Case and subtract 1 point for any ambiguity in terms of which button links to which screen	X : total is 6	X : total is 1			X	
Function Understandability	User able to understand what to do at each interface	Measure: Step through Use Case and subtract one point for ambiguity as to data to be entered on screen at any point	X : total is 5	X : total is 1			X	
Usability Compliance	Comply with requirements for interface design	Measure: More than one task required on a single interface. Subtract one from total and divide by total.	X: total is total no interfac es	X: total is total no interfac es			X	
Portability Metrics								
Hardware environmental adaptability	Adapt to different status of system : hardware link exists or not	Measure: Any acknowledgement of System online or offline on Interface OR DFD			X (0 or 1)	X	X	
System software environmental adaptability	Tailoring for user's preferred interface	Measure: Any functionality in interface for selecting which data to view.			X (0 or 1)		X	
Maintainability Metrics								
Scope	Design should be extendable to entire system	Measure: Context diagram deals with Tracking as well as Planning			X (0 or 1)	X		

Interconnectivity	Design should not repeat similar operations	Measure: Level 0/1 diagrams should link Use Cases through processes on the Risk Database			X (0 or 1)	X		
Abstraction	Design should be abstract at the higher level	Measure: Context diagram uses generic data flows			X (0 or 1)	X		

Table 3: Guidelines for the Planning Module Design Evaluation

			Scope of Measure				Design Product Measured		
The attribute of quality to be verified	**Goal : What to achieve in this area**	**Metric : How to measure this**	**Use Case 1**	**Use Case 4**	**Use Case 5**	**Overall**	**DFD**	**Interface**	**Data Table**
Functional Metrics									
Functional Adequacy	Whether all functionality exists	Interface functionality deals with forms/screens/reports/menus etc 1.1 Status screen shows Continue, Plan Advance or Plan Delay 4.1 Workpackage Task list has option to add new Task 4.2 Final Screen to display summary of Task data and submit (may be same as data entry screen) 5.2 Select 'Accept', 'Accept after modification' or 'Reject' after each quality attribute for each work product	X : 1	X : 2	X : 1			X	
		DFD should have processes for each data manipulation 1.1 Compare actual task time to planned time/duration 1.2 Save actuals with plan data 4.1 Retrieve data on tasks from Work Package 4.2 Retrieve data on people and material resources from Task 4.3 Retrieve data on time planned for task from Task 5.1 Retrieve list of completed Work Products with their Quality Attributes list	X: 2	X : 3	X : 1		X		
Total Score for Functional Adequacy Measure =Actual DFD/6 + Actual Interface/4									

Functional Implementation Coverage	Whether the functions are enough to do what the client wants	Interface: should show the screens/links. (Note: some of these may be in the same screen) 1.1 Status of project showing actuals against plans and estimates 1.2 Status screen should include Plan Advance or Plan Delay where necessary 4.1 Display List of Work Packages for that user account 4.2 Table to display and enter resources on task 4.3 Table to display and enter participant skills and skill level. 4.4 Table to display and enter materials and material quality and quantity 4.5 Table to enter Actual Time for task 4.6 Form to enter new task details 5.1 Display list of completed work products	X : 2	X : 6	X : 1			X	
		DFD deals with all data reports 1.2 Report Project Plan data 4.1 Report Work Package Data 5.1 Report Work Product Data	X : 1	X : 1	X : 1		X		
		Data Table stores all data attributes in data base 1.2 Project Plan has planned and actual times/duration 4.1 Work Package has task list	X : 1	X : 2	X : 1				X

		4.2 Task list has resources (people with skills and materials with quantity and quality) 4.3 Task list has the actual time linked 5.1 Work Products have quality attributes links for each product							
Total score for Functional Implementation Coverage Measure= Actual DataTable/4 + Actual DFD/3+ Actual Interface/8									
Access auditability	Records should be kept of participants contributions.	Measure: Exists: Login screen 1/3 Login process 1/3 User data attached to database data 1/3				X	X : 1/3	X : 1/3	X : 1/3
Access controllability	Access to system should be controlled to some data.	Measure: Exists Data link between login process and all report data for each screen in Use Case	X	X	X		X		X
Functional Compliance	DFD rules	Data Flows are maintained between levels				X	X		
		Data base consistency : only one process writes to each data base				X	X		
Usability Metrics									
Completeness of description	All functions described in an understandable manner	Measure: Step though Use Case and subtract one point for ambiguity as to data presented in screen at any point	X : total is 2	X : total is 6	X : total is 1			X	
Evident Functions	User able to understand how to perform tasks from design	Measure: Step though Use Case and subtract one point for ambiguity in terms of which button links to which screen	X : total is 2	X : total is 6	X : total is 1			X	

Function Understandability	User able to understand what to do at each interface	Measure: Step though Use Case and subtract one point for ambiguity as to data to be entered on screen at any point	X : total is 1	X : total is 6	X : total is 1			X	
Usability Compliance	Comply with requirements for interface design	Measure: More than one task required on a single interface subtract one from total and divide by total.	X: total is total no interf aces	X: total is total no interf aces	X: total is total no interf aces			X	
Portability Metrics									
Hardware environmental adaptability	Adapt to different status of system : hardware link exists or not					X	X	X	
	Adapt to different hardware of system	Measure: Any acknowledgement of System online or offline on Interface OR DFD				X (0 or 1)			
System software environmental adaptability	Tailoring for user's preferred interface	Measure: Any functionality in interface for selection which data to view.				X (0 or 1)		X	
Maintainability Metrics									
Scope	Design should be extendable to entire system	Measure: Context diagram deals with Tracking as well as Planning				X (0 or 1)	X		
Interconnectivity	Design should not repeat similar operations	Measure: Level 0/1 diagrams should link Use Cases through processes on the Risk Database				X (0 or 1)	X		

Abstraction	Design should be abstract at the higher level	Measure: Context diagram uses generic data flows				X (0 or 1)	X		
Re-Use	Design has re-used features from PPM	Measure: Each design should match PPM design in: a. Context Level outline b. Interface generic design c. Data Table attributes	X (0 or 1)	X (0 or 1)	X (0 or 1)		X		

Table: Guidelines for the Tracking Module Design Evaluation

For Hypotheses 4-5

In Hypothesis 4 and 5 the average actual time spent on the design activities will be compared for the control and treatment groups. From the time logs and number of design defects, the efficiency is compared for H4. The design defects are identified by the difference between the "perfect numeric value" identified in the measurement, formula and data elements computations column of [24] and the actual value calculated from the subject designs. For H5, time logs and number of designers involved form the basis for the cost effective calculations based on Net Present Value (NPV) will be compared.

For Hypotheses 6 - 12

A questionnaire to participants in the Pair groups will address issues with pair work experience as well as the quality of the design product produced according to the variables in hypotheses 6-12. Qualitative analyses of the questionnaire will be performed against the subjects design performance in pairs and as an individual.

Threats to Validity

The following types of validities are considered in the design of this experiment.

Internal Validity

In this experiment, subjects are divided into treatment groups of pairs and control groups of individuals for the first half of the experiment, and the two groups will be switched for the second half of the experiment. It can be argued that subjects who work as a pair in the first half will have an advantage of peer learning that put them in a better stead when they later work as individuals. Therefore, the analyses of the two groups will be done separately before they are combined.

Although the students will be randomly allocated to tutorial groups, they will however, be given the opportunity to choose a partner, in an effort to reduce personality clash issues between partners given the relatively short period of time for producing the deliverables. That choice might have a threat to conclusion validity if not all pairs have the same knowledge and competence in design work. Therefore, students' WAM's will be considered in the analyses to distribute the students according to their performance levels combinations.

Another threat to the internal validity is the varying guidance abilities of the tutor or lecturer overseeing each group during the different tutorial sessions. To reduce such impact, an on-line FAQ page of the subject course addressed all questions raised in all tutorial sessions for all subjects to access.

Construct Validity

The experiment was structured to measure the design products across two different work approaches. The designs are based on the subject's construction of DFD's, Databases and Interface diagrams. The measures of evaluation are identified and developed before the subjects start working on their designs to eliminate interpretation difference. The work approaches were individual work and pair work. The subjects are unfamiliar with pair work, so the questionnaire will be used to exclude partners who do not follow the pair work process. Therefore, there is no threat to construct validity.

Conclusion Validity

The conclusion validity of this experiment depends on three factors. Firstly, the statistical power is not considered threats since the participating subjects are undergraduate students and it is possible to have a large number of participants in comparison to other studies.

Secondly, the measures for the experiment outcomes are fairly objective. The efficiency and cost effectiveness are based on actual time logs. The measurements of the quality of the design outcomes are based on metrics derived from the ISO/IEC 9126 set of software product quality standards. This allows for quality characteristics to be evaluated through a tallying process that greatly reduces the subjectivity associated with measuring quality characteristics of products. In addition, one evaluator will conduct the evaluation and assessment of all design outputs to ensure consistency.

Finally, the abilities of the subjects do not change between paired or individual work approaches since the same subjects are allocated to both approaches at some stage. Therefore, the conclusion validity can be considered high.

External Validity

The subjects are undergraduate students; therefore, their selection may not be representative enough to directly transfer the results to the software industry. However, the subjects are in their pre-final year prior to joining industry, their design knowledge has been recently acquired and tested, and their task is to design part of a solution to a realistic problem. So, it can be argued that the results will have a high level of external validity. In any case, the results are of interest, as it is important to find out whether the quality of the products produced by pair work justifies the costs involved.

There is a great need for replication to enable the generalization of the results. The ER-Model may not be representative of industrial problems. The data model used in this study is smaller and less complex than industrial data models. However, the diagram uses a majority of the concepts normally found in an ER-model.

Conclusion and Future Analyses

The report presented the empirical design of an experiment to be conducted with 3rd year students in computer engineering. The experiment aims to assess the effect of pair work on the design phase of the software lifecycle. The objectives of the experiment are (a) to evaluate the quality of design deliverables produced by designers working in pairs compared to designers working individually, (b) to compare the efficiency and cost effectiveness of a design process carried out in pairs to an individually carried out design process. In addition, information about the practices and emotive of partners in pair work experience is obtained.

For the quality measures of the design products, assessments will be made according to quality characteristics, sub characteristics and metrics derived from the ISO/IEC 9126 software engineering product quality standards in order to reduce subjectivity. In addition, assessments of the design process will be based on efficiency and cost effectiveness. Pair work practices of partners will be analysed from data that will be obtained from participants in a questionnaire after the pair work experience.

Acknowledgements

Many thanks to Dr. Barbara Kitchenham, Cat Kutay, Mike Berry and Dr. Karl Cox for their valuable input and collaboration in this work.

References

[1] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures Methods and Case Studies*: Addison-Wesley, 2002.

[2] T. DeMarco, *Structured Analysis and system Specification*. New York, 1979.

[3] D. Budgen, *Software Design*. Essex, UK: Pearson Educational Limited, 2003.

[4] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," *IEEE Software*, pp. 19 - 25, 2000.

[5] A. Pandy, N. Kameli, A. Eapen, C. Miklos, F. Boudigou, I. Sutedjo, M. Paul, V. Vijay, and W. Mcdermott, "Application of Tightly Coupled Engineering Team for Development of Test Automation Software - A real World Experience," presented at 27th Annual International Computer Software & Applications Conference, Dallas, Texas, 2003.

[6] L. Williams and R. R. Kessler, "Experimenting with Industry's "Pair Programming" Model in the Computer Science Classroom," *Journal on Software Engineering Education*, 2000.

[7] J. T. Nosek, "The Case for Collaborative Programming," *Communications of the ACM*, vol. 41, pp. 105 -108, 1998.

[8] J. Nawrocki and A. Wojciechowski, "Experimental Evaluation of Pair Programming," presented at European Software Control and Metrics Conference ESCOM 2001, Maastricht, 2001.

[9] P. Freeman, "Fundamentals of Design," in *Tutorial On Software Design Techniques*, P. Freeman and A. I. Wasserman, Eds., 4th ed: IEEE Computer society press, 1984.

[10] C. Easteal and G. Davies, *Software Engineering: Analysis and Design*. London: McGraw-Hill Book company, 1989.

[11] I. Sommerville, *Software Engineering*, 4th ed: Addison-Wesley Publishers Ltd., 1992.

[12] E. Robertsson and H. Eriksson, "An Empirical Study on product and Process quality in object-oriented Design," in *School of Information Systems*. Sydney: University of New South Wales, 1999.

[13] R. S. Pressman, *Software Engineering, A Practitionar's Approach*, 2nd ed: McGraw-Hill Inc., 1987.

[14] E. Isaacs and A. Walendowski, *Desiging from both sides of the screen*: New Riders publishing, 2002.

[15] B. Curtis, H. Krasner, and N. Iscoe, "A field Study Of The Software Design Process For Large Systems," *Communications of the ACM*, vol. 31, pp. 1268-1287, 1988.

[16] L. Williams and R. R. Kessler, *Pair Programming Illuminated*: Pearson Education, Inc., 2003.

[17] L. Williams and R. R. Kessler, "All I really need to know about Pair Programming I learned in Kindergarten," *Communications of the ACM*, vol. 43, pp. 108 - 114, 2000.

[18] P. R. Laughlin, R. P. McGlynn, J. A. Anderson, and E. S. Jacobson, "Concept attainment by individuals versus co-operative pairs as a function of memory, sex and concept rule," *Journal of Personality and Social Psychology*, pp. 410--417, 1968.

[19] K. Beck, "Embracing Change with Extreme Programming," *IEEE Computer*, pp. 70 - 77, 1999.
[20] R. Jeffries, "What is Extreme Programming," vol. 2002, 2001.
[21] A. Cockburn and L. Williams, "The costs and benefits of pair programming," in *Extreme Programming Examined*: Addison Wesley, 2001.
[22] L. Williams and R. Kessler, "The effects of "Pair-pressure" and Pair-Learning" on Software engineering Education," presented at Conference of Software Engineering Education and Training, 2000.
[23] J. C. Jones, *Design Methods: Seeds of Human Futures*: John Wiley & Sons, 1981.
[24] ISO/IEC, "ISO/IEC 9126-3 Software engineering -Product quality- part3: Internal metrics," 2002.
[25] ISO/IEC, "ISO/IEC 9126-1 Software engineering- Product quality- Part 1: Quality model," 2001.
[26] B. Wong, "An Investigation of the Cognitive Structures used in Software Quality Evaluation," in *Information Systems, Technology and Management*. Sydney: University of New South Wales, 2002.
[27] ISO/IEC, "ISO/IEC 9126-2 Software engineering -Product quality- part2: External metrics," 2002.
[28] ISO/IEC, "ISO/IEC 9126-4 Software engineering -Product quality- part4: Quality In Use metrics," 2002.
[29] K. Beck, *eXtreme Programming explained, Embrace Change*: Addison-Wesley, 2000.
[30] A. Cockburn, *Writing effective Use Cases*: Pearson Education Corporate Sales Division, 2001.