Energizing Software Engineering Education through Real-World Projects as Experimental Studies

Jane Huffman Hayes
Computer Science Department
Laboratory for Advanced Networking
University of Kentucky
Lexington, KY 40506-0046 USA
and Science Applications International Corporation
hayes@cs.uky.edu or jane.e.hayes@saic.com

Abstract

The use of a semester-long project to apply theoretical knowledge and provide "hands-on" experience has long been a staple of software engineering courses. Our experience shows that a typical industrial project can also enhance software engineering research and bring theories to life. The University of Kentucky (UK) is in the initial phase of developing a software engineering curriculum. The first course, a graduate-level survey of Software Engineering, strongly emphasized quality engineering. Assisted by the UK Clinic (part of the UK Medical School), the students undertook a project to develop a phenylalanine milligram tracker. It helps phenylketonuria (PKU) sufferers to monitor their diet as well as assists PKU researchers to collect data. The project was also used as an informal experimental study. The applied project approach to teaching software engineering appears to be successful thus far. The approach taught many important software and quality engineering principles to inexperienced graduate students in an accurately simulated industrial development environment. It resulted in the development of a framework for describing and evaluating such a real-world project, including evaluation of the notion of a user advocate. It also resulted in interesting experimental trends, though based on a very small sample. Specifically, estimation skills seem to improve over time (with as little as one experience) and function point estimation may be more accurate than LOC estimation.

1. Introduction

Software engineering courses have long used semester-long projects to apply theoretical knowledge and provide "hands-on" experience. Generally, students are grouped into teams that perform all the activities of the software system development lifecycle and deliver a finished product at the end of the semester. Such an approach can be expanded to include experimental research studies. Learning can be enhanced by having teams undertake important, real-world projects that "double" as experimental studies.

1.1. Real-world project – phenylalanine tracker

The University of Kentucky (UK) is developing a new software engineering curriculum. The first course, a graduate-level survey of Software Engineering, was offered this past semester. The course emphasized software verification and validation and quality/reliability

engineering (we use the term reliability engineering informally). For example, students were required to perform formal technical reviews, to collect data for reliability growth models, and to perform extensive coverage testing. The students undertook a real-world project to develop a phenylalanine (phe) milligram tracker. The product, developed to run on a personal digital assistant (PDA), allows phenylketonuria (PKU) disease sufferers to monitor their diet as well as assists PKU researchers to collect data. The project was also used as an experimental study, testing numerous software engineering hypotheses.

Working together with the UK Clinic (part of the UK Medical School), the student's projects will be used in a pilot study to determine if such an application can advance PKU research and/or assist PKU sufferers. About one in every 15,000 infants born in the United States has the inherited (genetic) disorder PKU. People who are born with PKU are normal in every way except that to stay healthy they must follow a strict diet. The diet limits phe, a common part of most foods. If phe levels in the blood of a PKU sufferer stay too high for a long time, the damage to the brain is severe and irreversible [5].

The important, real-world project/study approach to teaching software engineering has been successful thus far. It helped to motivate the teams and to encourage development of higher quality products by the teams. The teams took seriously the importance of the problem that they were helping to solve. The approach taught inexperienced graduate students many software engineering and software verification and validation principles that they are applying to newly gained jobs and/or subsequent courses.

1.2. Paper organization

The paper is organized as follows. In Section 2, related work in software and quality engineering education is discussed. Section 3 describes the real-world project/study concept as well as a framework for describing and evaluating such course projects. Section 4 discusses the semester-long PKU project and experimental study, the notion of a user advocate, and describes the interaction with the Medical School and PKU sufferers. Benefits of the important, real-world project/study approach are presented in Section 5. Finally, Section 6 presents conclusions, directions for future work, and suggestions for how one might administer a software engineering course using this approach.

2. Background and related work

In this section, related work in the area of quality engineering education is presented. Section 2.1 discusses software engineering education literature and meetings. Section 2.2 presents information on other academic institutions that are pursuing software quality engineering education, with an emphasis on their offered courses (particularly if a project component exists).

2.1. Software engineering education

Software engineering education has been in the spotlight recently, with much progress on the Software Engineering Body of Knowledge (SWEBOK) [6] as well as on what constitutes an appropriate (and in the future accreditable) curriculum for software engineering [1]. There are many articles in the literature on techniques/approaches being used to teach software engineering and computer science, and entire conferences devoted to this subject such as the Conference on Software Engineering Education and Training (CSEET), Frontiers in Education (FIE), and the SEI Software Engineering Education Conference. The International

Conference on Software Engineering (ICSE) also invites papers on the topic, and in June, six papers on the subject (roughly 10% of the total number of ICSE regular papers) were presented [4].

2.2. Reliability and quality engineering education

It is more difficult to find information on software quality/reliability engineering education, except as a subset of software engineering education. There is quite a bit of activity in the area though, as evidenced by the many websites offering information on reliability engineering, reliability engineering courses, etc. For example, the University of Twente offers a reliability engineering course titled "Total Quality for Software Engineering." The curriculum consists of four one-week sessions. Lecture topics include Introduction to Software Reliability, Software Reliability Improvement, Total Quality Software Management, Software Reliability Engineering Practice. Students also performed lab work, but no project [15].

A web-based software quality assurance (SQA) course was developed by the University of Central Florida (UCF), working with Warsaw University of Technology and Delft University of Technology [14]. The University of California-Irvine is undertaking a project to develop an environment in which commercially-representative software engineering experiences can be simulated, called Software SimCity. They hope to use the environment to teach the cause and effect of the critical decisions that are typically made during the software life cycle [9], including reliability decisions.

The SEI has developed a suggested software engineering curriculum [1]. It recommends a series of courses, including Introduction to Software Engineering, Software Construction and Evolution, and Software Design Project. These three proposed courses would discuss quality and reliability engineering to varying degrees of depth and have some project component (some individual, some team-based) [1].

3. Real-world project/study – concept and framework

In developing this course, the instructor: (a) interacted with a number of colleagues in the software quality/reliability engineering field such as Dr. Jeff Offutt of George Mason University,(b) researched the syllabi of many software engineering and software reliability courses [3,7,8,12], and (c) spoke with the largest software engineering employers in Lexington. This information, together with personal experience, dictated the course layout as well as the project layout and content.

3.1. Project concept

The goals of the project were to: (a) teach software quality engineering concepts to students, (b) give students an opportunity to apply software quality engineering concepts in an industry-representative real-world project, (c) develop a framework for using projects as experimental studies, and (d) build a product to fill an important need.

To meet these goals, we defined a project with many facets: (a) compelling real-world project to capture and hold the student's attention for an entire semester, (b) "easy" to implement project so students could concentrate on the software and reliability engineering process as opposed to subjecting them to "death by coding", (c) phased implementation to encourage design/implementation for maintenance/enhancement (commercially-representative), (d) end users "in the loop" to motivate students, (e) demonstrations used to

encourage quality, (f) presentations required throughout the semester to ensure strong management of the project as well as to closely resemble a real development environment, (g) estimation and later collection of detailed metrics required to monitor quality (to convince students that formal technical reviews (FTRs) are worthwhile as well as to instill a sense of process), (h) students required to evaluate each other's work using 360-degree feedback forms and presentation evaluation checklists, (i) weaker teams "steered" at each phase of the project by giving them "best of breed" samples, (j) team progress and timely instructor feedback assured by having homework "feed" the project phases, and (k) concept of a user advocate (or 'voice of the user') and how it might impact development investigated.

3.2. Project framework

While examining the above information and pondering the use of a course project as an experimental study, it became apparent that projects have certain aspects. These aspects or characteristics can serve as a framework for structuring, describing and evaluating any project. The aspects provide a scheme that can be used to understand projects, compare projects, or evaluate projects and look for areas of improvement. To develop such a scheme, the Basili, Selby, and Hutchens [2] framework for experimentation, addressing many of the aspects of a course project, was used as a departure point. There are advantages to doing so. It ensures that our framework is in keeping with published, well-grounded work. It may encourage instructors to use course projects for more than just student grades (to apply experimental software engineering principles and use projects as part of their research). We enhanced this framework by adding parts to phases and by adding levels to many of the parts (see italics in Figure 1). The resulting course project framework, summarized in Figure 1, designates four phases: (1) definition, (2) planning, (3) realization, and (4) interpretation. Each of these phases is discussed next.

3.2.1. Project definition*:* Definition refers to the project definition phase, the time when an instructor decides the scope and objective of the project. There are eight parts to the definition phase: motivation, purpose, object, perspective, domain, scope, importance, and end user. Just as with experimental studies, there can be many motivations, purposes, or objects in reliability engineering course projects [2]. In addition, there can be several scopes, end user classes, and importance levels. For example, the motivation of a project may be to understand, learn, or validate the effect of a certain technology. The purpose of a project may be to test an existing system, to implement a domain-specific application, to evaluate the effectiveness of design processes, where the "object" of the project may be the final software product, a development process, etc. Though most projects are from the perspective of the instructor, they may be from many other perspectives such as tester, customer, and/or user advocate.

The domains that typically comprise projects are individual engineers or teams (software or reliability engineers or teams thereof) and programs on which teams or engineers work. Basili et al classify experimental study scopes by looking at the size of the domains considered [2], as does this project framework. Projects that are blocked subject-project examine one or more objects across a set of teams and a set of programs. Projects that are replicated project scope look at objects across a set of teams and a single program. Multiproject variation projects examine objects across a single team and a set of programs. Projects that are single project scope look at objects on a single team and a single program. A course project can be characterized as having safety-critical importance (potential loss of

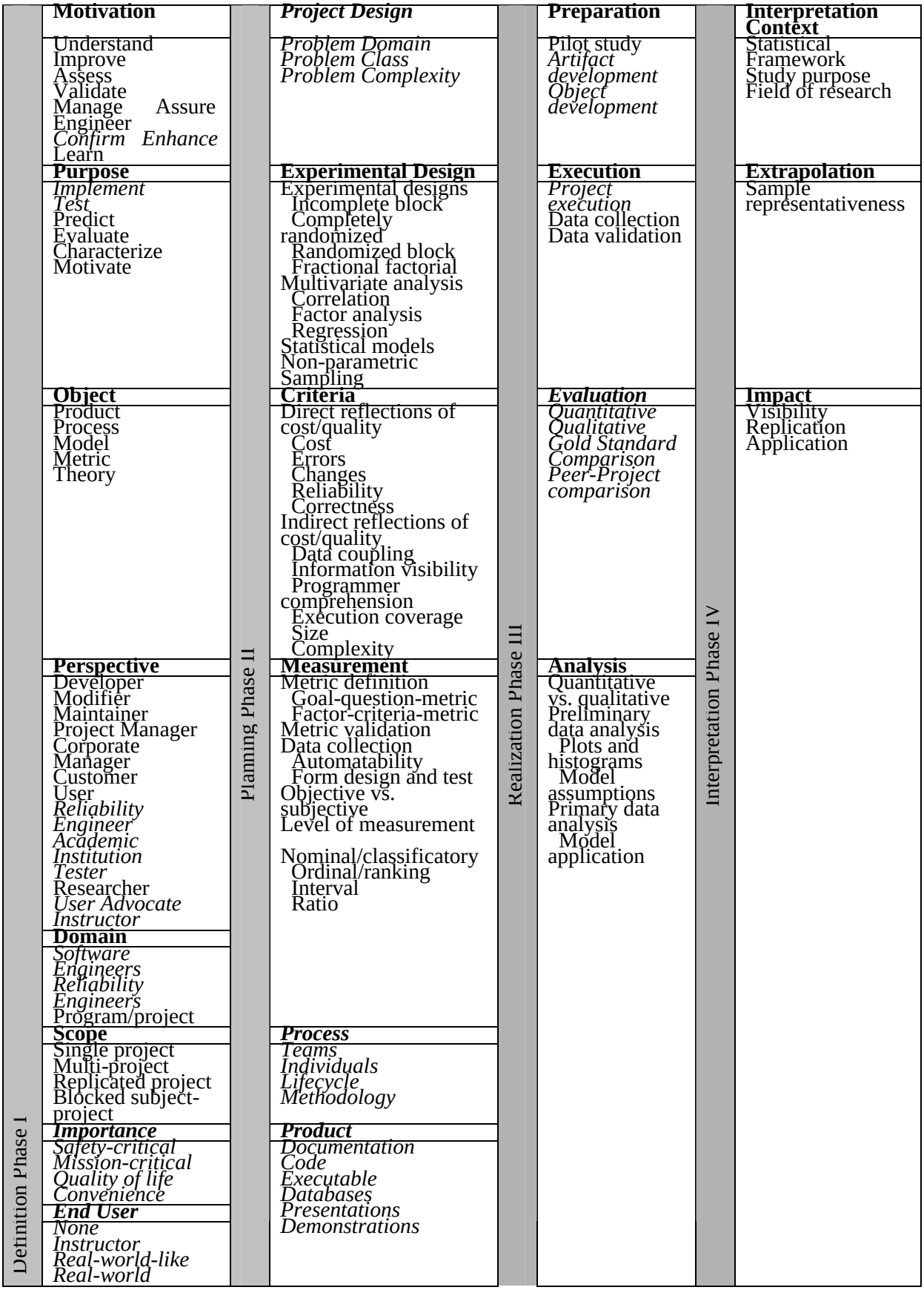

Figure 1. Summary of the framework for course projects.

human life), mission critical importance, quality of life importance, or convenience importance. The end user of the course project can be categorized as none, instructor, real-world-like, or real-world.

3.2.2. Project planning*:* Planning refers to the project planning phase, the time when an instructor designs the project and/or experiment. There are six parts to the planning phase: project design, experimental design (optional), criteria, measurement, process, and product. Project design encompasses the problem domain (such as financial, transportation, defense, education, real estate, insurance, etc.), the problem complexity, and the problem class (such as payroll, air traffic control, etc.). The experimental design is optional, but recommended if an instructor is engaged in research and wants to experimentally study an aspect of software or quality engineering as part of the project. The experimental design, criteria, and measurement components are outside the scope of this paper but are covered in detail elsewhere [2]. The planning process part encompasses teams (by size), individual programmers, use of a standard process, and use of a particular lifecycle or methodology. Finally, the planning product part covers documentation, code, executable, databases, presentations, homework, and demonstrations

3.2.3. Project realization*:* The project realization phase is the time when the software engineering students accomplish the project and/or experiment. There are four parts to the realization phase: preparation, execution, evaluation, and analysis (optional). For an experimental study, preparation often includes a pilot study [2]. In software engineering course projects, preparation may include preparation of project artifacts, development of code (if the project involves testing), etc. Execution covers the actual project accomplishment by students as well as data collection and validation (if also an experimental study). Evaluation refers to the review of the completed project for grade assignment. It includes qualitative evaluation, quantitative evaluation, as well as comparing the projects to each other and/or to a gold standard. The analysis component applies if the project is also an experimental study. It is outside the scope of this paper but is covered in detail elsewhere [2].

3.2.4. Project interpretation*:* Interpretation refers to the project interpretation phase, the time when the instructor and/or researcher derives a result from the experimental study/project. There are three parts to the interpretation phase: interpretation context, extrapolation, and impact. These components are outside the scope of this paper but are covered in detail elsewhere [2].

3.3. Contributions

Our concept went beyond those described in Section 2 in several ways: it focuses on important, real-world projects; it does not have predefined problem and/or solution sets (making it harder to grade, but driving the students to extend their decision making skills); it suggests the idea of “double dipping” and using course projects as experimental studies on a routine basis; it introduces the notion of a user advocate; and it suggests facilitating these ideas through the use of a flexible project framework that builds on the work of Basili et al [2]. Our work enhanced their framework by adding numerous parts and levels specific to course projects that are doubling as studies. Our course project concept is similar to those of software engineering and quality engineering education in that a semester-long project was assigned, and that teams undertook these projects. Another similarity is that students were required to read many of the same reference materials.

4. Phenylalanine (phe) tracker – the sample project

To examine the sample project undertaken in the UK software engineering course, the project framework defined above is used. The notion of a user advocate is presented. Next, samples of the artifacts and object are presented. Analysis results from the experimental study aspect of the project are discussed. A discussion of interactions with the UK Medical School rounds out the section.

4.1. Classification of Phe Tracker project

This section examines the PKU project more closely, using the framework described above. Our *motivation* was to engineer a product for easy modification as well as to better understand the maintenance process. We undertook a project whose *purpose* was to predict the size and effort to build the application (the product, i.e. object) as well as to *implement* the problem solution. We did so from the perspective of the *developer, maintainer,* and *user advocate*. The product was examined in a replicated project study (*scope*), where 33 software engineers, student through professional (from the software engineer *domain*), developed one software system (from the program/project *domain*). It was quality of life *importance* with real-world *end users.* The software system developed belonged to the medical problem domain in the nutrition monitoring system problem class *(project design)*. It was developed with no specific *experimental design.*

Objective *measurement* of the engineering processes was in several criteria areas: size estimation effectiveness, complexity, fault detection, reliability growth, the relationship of design characteristics to maintainability, the relationship of maintainability to reliability. *Preparation* included artifact development (narrative statement of scope) and object development (development lifecycle steps to be followed), and *execution* was broken into three phases and incorporated manual monitoring of activity. *Evaluation* included the application of qualitative criteria and peer-project comparison. Analysis, interpretation context, extrapolation, and impact are still being finalized, but initial results are available.

4.2. Role of user advocate

Before presenting the object and artifacts of the Phe Tracker project, it is important to understand an important concept. In setting up the project, the instructor/researcher sought to find an important, real problem that could be "tackled" in a one-semester project. The instructor also wanted the students to be able to perform requirements elicitation, a very difficult but most important step. To facilitate course schedules and to examine the idea of a *user advocate*, the instructor served as the end user. However, the instructor does not have PKU but rather has detailed knowledge of the disease and its management through eight years of interaction with a PKU sufferer.

The instructor had been discussing the Phe Tracker idea with the PKU patient for several years and was able to serve as the end user in the initial requirements elicitation session. After an initial product capability was delivered by the students, it was demonstrated to the PKU patient as well as the UK Medical School who helped direct the second requirements elicitation session. We found that the user advocate notion worked well and that it improved requirements elicitation and therefore improved the quality of the final product. We plan to empirically study this in the future.

4.3. Phe Tracker project - object and artifacts

Acting as user advocate, the instructor (and students) held the first requirements elicitation session. The resulting narrative statement of scope is shown in Figure 2. Students were grouped into eleven 3-person teams. For the first phase of the project, the students were instructed to use the software engineering lifecycle described in Pressman [15] and to prepare the artifacts listed in Figure 3. Note that the teams had the option of using structured (SA) or object-oriented (OO) analysis for their project.

Customer Description of Problem for Homework #1

I want the application to run on my workstation (Windows) and on my Palm Pilot (Palm OS)
I want it to count my protein gram intake each day
I want to be able to tell it how many protein grams I should have each day – a daily protein budget
I want to be able to choose from a menu of different food types (like fruit, vegetable, meat, dairy, etc.) and see a detailed list of food choices.
So if I clicked on fruit, I might see an alphabetized list:
Apple
Banana
Cherry
Etc.
I then can click on the food that I ate (such as Cherry) and it would look up the protein amount for that food (in a table that is already provided, the user does not enter this information)
and add it to my daily total of protein grams I had eaten (and subtract it from my daily budget).
I want to be able to display my daily total of protein grams, how much I have left of my daily budget of protein grams,
as well as a weekly average (take the last 7 day totals and divide by 7) of protein grams.
The display of these things should be available on the Palm Pilot or workstation.
Also, on the workstation, I want to be able to get a printed report of these 3 things.
You can choose what programming language you want to use.
If I have not answered a question you have, make it up and document it as an assumption. For example: I assume that there is enough memory on a Palm Pilot to hold the protein look-up table.

Figure 2. Problem statement from first requirements elicitation.

After Phase I was completed, the instructor evaluated the projects and returned those results along with samples of excellent solutions. Phase II was assigned the day that Phase I was returned. The artifacts required differed based on the methodology (SA or OO) used. Our undertaking cannot be considered a strict experimental study because of this flexibility, because students were given "best of breed" examples at the end of each phase, because the teams could choose their programming language and environment, because there was no specific experimental design, and because there was no attempt to control for threats to validity (internal or external).

The resulting project applications were impressive. One team, Team 8, developed a product as opposed to just a program, complete with installation disks, context-sensitive help, a professional interface and logos, etc. Their product logo (displayed during the Setup process) is shown in Figure 4. After Phase II, Phase III was assigned. This phase consisted of making several major modifications to the Phase II program. Another requirements elicitation session followed and resulted in a modified narrative statement of scope.

As can be seen from the discussion above, a significant amount of time was invested in preparing the project assignments and interacting with the students throughout the project. In addition, grading the projects was quite time consuming. The instructor read through each project quickly, ensuring that all required artifacts had been delivered. The instructor then went back through each artifact to determine completeness, attention to detail, consistency, etc. Using initial grading criteria as a basis, the instructor built a point deduction system for each artifact (for example, subtract 1 point if data flow diagrams fail to show the daily PKU intake being stored for subsequent computations). Each project was reviewed numerous times to ensure consistent grading.

1. Planning and Estimating
  a) develop a statement of the scope of the problem (narrative) – sections 3.3., 5.3 Pressman
  b) develop a high level problem decomposition (your choice how to represent) – sections 3.3.2, 5.6 Pressman
  c) develop a size estimate (either in LOC or FPs, should look like figure 5.3 or 5.4) – section 5.6 Pressman
  d) Develop a risk table for this project – Chapter 6 Pressman (should look like Figure 6.2 but sorted with worst risks at the top)
  e) *Calculate a task set selector value for building this system (casual, strict, or structured) – section 7.3 Pressman (should look like Table 7.2)*
2. Analysis
  a) Develop a system context diagram for this system – section 10.6 Pressman (should look like Figure 10.6)
  b) IF doing Structured Analysis, then:
    a. ERD
    b. DFD (level 0, 1)
    c. CFD
    d. Hold a FTR and show results
  c) IF doing Object-Oriented Analysis, then:
    a. Use-Cases
    b. Object-Relationship model
    c. Object-Behavioral model
    d. Hold a FTR and show results
3. High Level Design
  a) IF doing Structured Analysis, then:
    a. DFD (level 2,3)
    b. Program Structures
    c. Hold a FTR and show results
  a) IF doing Object-Oriented Analysis, then:
    a. Object interaction model (collaboration model in Pressman, Fig. 22.4, 22.5, also Fig. 12.13 Schach and Fig. 14.16 Sommerville)
    b. Detailed class diagram (Figure 12.14 Schach)
    c. Hold a FTR and show results
4. Presentation of Results to Class (2/22)
    ° Each team will have 5 minutes to present their results
    ° You may use viewfoils (transparency slides), PowerPoint slides, or posters
    ° Your presentation must present the major aspects of Phase I (you may not have enough time to show all the items developed, so some results may be combined)
    ° You will also be presenting the results of Phase II and III to the class later in the semester. Make sure that each team member speaks during one of the three presentations

Figure 3. Phase I project assignment and required artifacts.

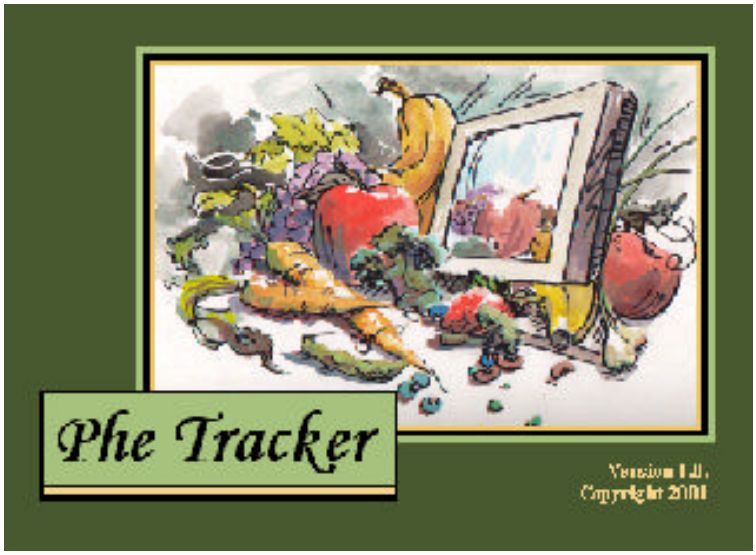

Figure 4. Team 8 product logo.

Artifacts of the lifecycle were also quite impressive. A level 2 data flow diagram for the Phase III project of Team 3 is shown in Figure 5. A partial list of the data gathered during each phase is listed in Table 1.

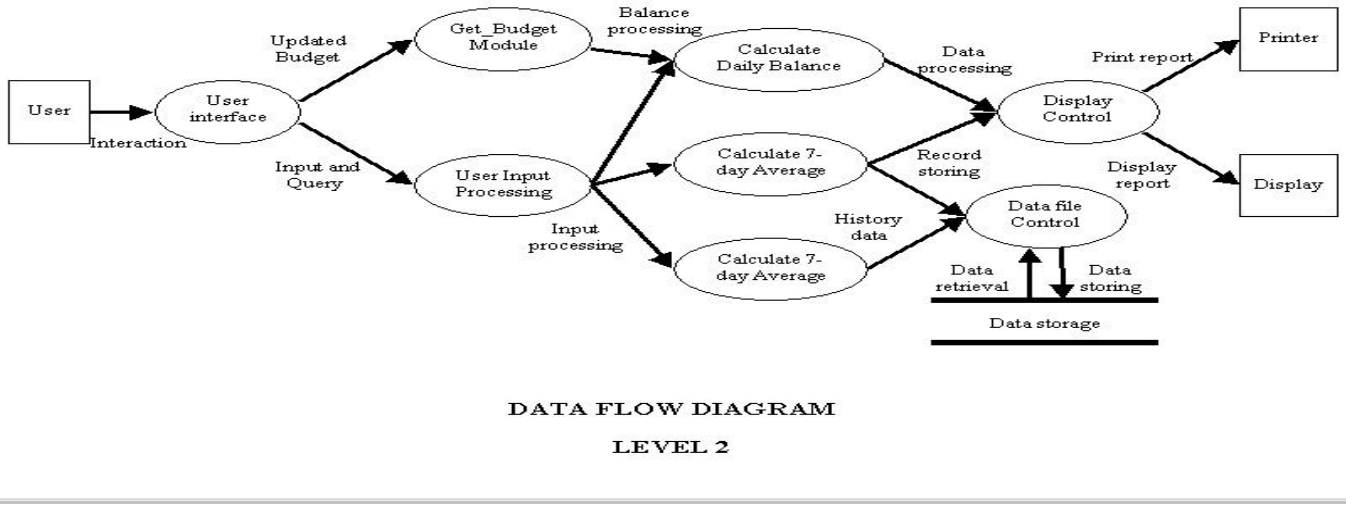

Figure 5. Team 3's level 2 data flow diagram for phase III.

4.4. Preliminary analysis of the Phe Tracker experimental study

The project is classified as an experimental study in section 4.1. There were three main hypotheses being evaluated by this small study: (1) maintenance on a product decreases its quality, (2) students will improve their estimating skills over time, and (3) programming language or paradigm has no impact on quality.

As the real-world project/study approach was just evolving during the CS 650 course, the experimental study was not handled with rigor or formality. For example, students selected their own teams (no controlling for effect of some teams having more experienced or talented

members than others), there were only 11 teams (small sample), and teams were permitted to select the programming language (nine selected Java, one selected C++, one selected VBasic). Also, many teams did not supply all requested data for all phases of the project. The students all worked to the same specifications and deadlines. Recall that implementation occurred in Phase II with modifications/maintenance in Phase III.

Table 1. Sample of type of data gathered during Phe Tracker project.

Name of Class or Method	Response for Class	Number of operations
Weighted Methods per Class	Number of Children	Number of attributes
Cyclomatic Complexity	Module Coupling Indicator	Number of messages sent
Lack of Cohesion in Methods	System Complexity	Specialization index
Depth of Inheritance Tree	Data Complexity	Coupling between Objects

Examining the first hypothesis, we looked at several sub-hypotheses: weighted methods per class (WMPC) will increase after maintenance occurs, complexity will increase after maintenance occurs, number of defects will increase after maintenance occurs, coupling will increase after maintenance occurs, cohesion will decrease after maintenance occurs, and size will increase after maintenance occurs. Our small study found no such trend for WMPC (3 teams had higher values after maintenance, 4 did not), complexity did increase (but only 3 teams reported data before and after maintenance), number of defects decreased for maintenance (all 4 teams reporting Phase II and III data had less defects in Phase III than in Phase II), and coupling was inconclusive (increased for 2 teams, decreased for 2 teams) as was cohesion. There were more function points (FPs) and lines of code (LOC) after maintenance (4 of 6 teams had an increase in FPs, 6 of 7 teams had an increase in LOCs). With such mixed results, we cannot say anything conclusive from our study about the relationship between maintenance and quality.

Looking at the second hypothesis, we had two sub-hypotheses: FP estimates will improve over time, and LOC estimates will improve over time. Our study found that 10 of the 11 teams improved their FP or LOC estimating, Error 3 was smaller than Error 2. Error 2 is the estimation error of teams in Phase II of the project (number of FP or LOC difference between estimated and actual as a percentage of the estimated value). Error 3 is the estimation error of teams in Phase III. This finding suggests that estimation skills will improve with experience. A related hypothesis was that students using FPs would make better estimates than those using LOC. Indeed, this appeared to be the case. All 6 of the teams reporting FP estimates and actuals were very accurate in estimating, as shown in Figure 6. Also, all FP actuals were equal to or larger than the estimates. For LOC, 2 teams underestimated and 4 teams overestimated with several teams missing their estimates by 100%.

For the fourth hypothesis, we had several sub-hypotheses: Java applications will have lower complexity than non-Java applications, Java applications will have lower coupling than non-Java applications, Java applications will require less effort, Java applications will have less defects. Unfortunately, we lacked the necessary data on the non-Java applications to examine these ideas.

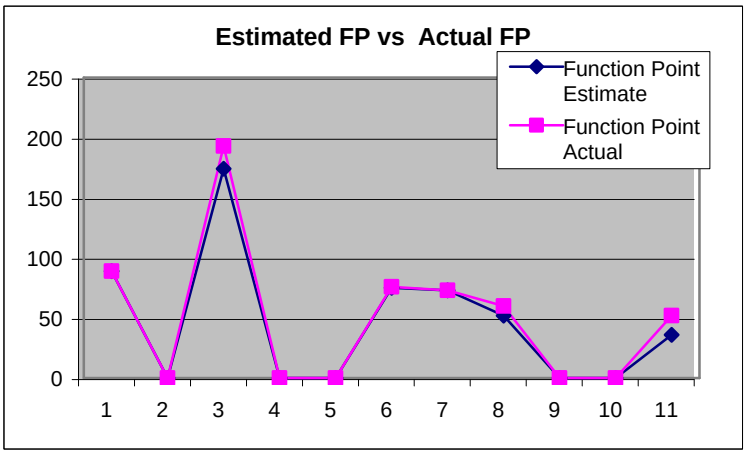

Figure 6. FP Estimation Error.

4.5. Interaction with Medical School

The interaction with the UK Medical School was to commence in late January/early February, but due to several reschedulings did not occur until mid-March. The first meeting involved Dr. Charlton Mabry, Linda Brooks, and Carol Reid of the Pediatriatic Endocrinology unit of the UK Medical Clinic, as well as a PKU patient and parent. All three of the UK Med School staff possess decades of experience diagnosing and treating PKU. During this interaction, two major requirements for the project were added (3-day diet history and ability to enter weight of food portions consumed).

The next interaction occurred in late-March. The Medical School personnel helped the instructor get in contact with Ross Laboratories to investigate their phenylalanine food database. Discussions with Ross Laboratories helped the instructor to generate a phenylalanine database from the USDA's website. In mid-April, the UK Medical School personnel were given a demonstration of the Phase II projects. They made a number of suggestions for improvements (e.g., also categorize foods alphabetically as many children do not know their food groups).

The Phase III projects were demonstrated to them in early May. Early in the Fall term we plan to work on a joint grant proposal to pursue this project further (to expand it to a palm-held computer, to possibly investigate monitoring of phe blood levels, etc.) for PKU research as well as for software quality engineering research (use this as a research bed). The Medical School also invited us to attend a PKU camp to demonstrate the project to many PKU patients and their parents. Note that throughout the project we consulted with one PKU patient, demonstrated projects to that patient, and used the feedback to improve the requirements and requirements elicitation process. Also, the patient attended our class in mid-April.

5. Assessment of success

At this point, it appears that the project was a success. There were numerous benefits provided to numerous groups of people. First, the students benefited from many software industry-representative experiences such as requirements elicitation, requirements changes, major enhancements or modifications to a program, development of a project that will enhance the quality of life of real users. The students learned many important software quality engineering concepts and were convinced of the need to build quality in as opposed to testing quality in. The students seemed to learn more and work harder to ensure program

quality because the project was important. The students were able to work on a project in a highly accurately simulated real-world development environment. This will benefit them greatly upon graduation and/or in later courses. Many students commented on how much they enjoyed doing something to help PKU patients, that it motivated them, and made them "feel good" to participate in an important, worthwhile project. Second, the end user benefited by receiving an application to help monitor and track phe intake (no such application existed previously). It appears that this will enable some patients to actually eat more while on this very restricted diet, thus improving their quality of life. The end user also gained some insight into the software engineering process as well as learned more about computers in general.

The Medical School benefited from this project by gaining a potential tool to help them track patient's intake of phe as well as to help their patients have higher quality of life. It may help researchers and disease management experts make useful discoveries by allowing them to carefully monitor/track phe intake. It also helped enhance their understanding of software and reliability engineering and computers. The instructor also felt very motivated and rewarded by undertaking this important project to fill a real need in the lives of PKU patients. In addition, the instructor had a chance to interact with PKU patients, the UK Medical School, and to empower a class of students to tackle an important project. The academic institution benefits since the students taking this course will be a notch above the students who take a typical software engineering or reliability engineering course. Also, there is a chance that this project will become a product that can be "donated" to the medical community and patients or technology that can be transferred to the commercial sector.

6. Results, conclusions, and future work

We realize that we have only examined one project and one use of the real-world project/study approach, further objective evaluation must be performed. With this caveat, what do these results mean to us? To instructors or researchers in the field of software engineering, they indicate that important, real-world problems are the best ones to assign as course projects. It means that these projects can be used as experimental studies also, with advanced planning and careful attention to the framework of the study/project. It means that preparing the project assignment and grading the projects will require more time, but the benefits will be worth it. To end users and medical researchers with interest in a useful, reliable product, it means that some investment of time to assist with requirements elicitation and to evaluate prototype demonstrations is worthwhile.

In addition to having students evaluate the course and the project component of the course, the instructor evaluated these items to extract "lessons learned." Suggestions to someone else implementing a software engineering course are:

- select an important, real-world project and have it double as an experimental study
- ensure that the implementation aspect is simple, allow students to concentrate on reliability engineering and not "coding"
- plan to have the students make a major modification to the project to ensure design for maintainability
- if real end users are not available, consider making yourself a user advocate
- do not underestimate the amount of time required to prepare such a project or to evaluate each phase of the project
- try to collaborate with another department or school at your institution.

In conclusion, the important, real-world project/study approach to teaching software engineering has been successful thus far. It helped to motivate the teams, to increase team

interest in the project, to inspire greater effort on the part of the teams, and to encourage development of higher quality products by the teams. The teams were highly committed to the success of the project as they understood and took seriously the importance of the problem that they were helping to solve. The approach taught inexperienced graduate students many important software reliability engineering principles. In addition, the project accurately simulated an industrial development project that also served as an informal experimental study.

There is further work to be done though. First, the project framework needs to be evaluated and enhanced through use by other instructors/researchers. Second, the success of the important, real-world project/study concept needs to be quantified. This will be facilitated by further analysis and interpretation of the results of the Phe Tracker project. Third, the important, real-world project/study concept needs to be validated by other instructors/researchers. Finally, the Phe Tracker project needs to be continued. Future plans include: porting the application to a PDA (in progress), enhancing the application to allow synchronization of phe intake data logs, running a pilot study of the phe application (with the Med School), gathering requirements as a result of the pilot study, and writing a grant to receive research funding for both the medical aspects and software/quality engineering aspects of this research bed and project.

Acknowledgments

I would like to thank all of the students who participated in the Software Engineering course (UK's CS 650). Thanks to Team 8 (Andrew Mertz, Venkata Velagapudi, Sanjiv Ganguli, and Jessica Mertz - artist) and Team 3 (Jim Carigan, Ming Jiang, Xiangyang Qin) for providing artifacts, and to Naresh Mohamed for assisting with analysis. A special thanks to Hannah and Kelly Marcum and to the UK Medical School for their participation in this project.

References

[1] Bagert, D. Hilburn, T., Hislop, G., Lutz, M., McCracken, M., and S. Mengel. Guidelines for Software Engineering Education Version 1.0. Technical Report CMU/SEI-99-TR-032, Software Engineering Institute, Carnegie-Mellon University, October 1999.

[2] Basili, V., Selby, R., and D. Hutchens. Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 47 July 1986, pp. 733-743.

[3] Bullard,C., Caldwell, I., Harrell,J., Hinkle, C. , and J. Offutt.. Anatomy of a Software Engineering Project. In *Proceedings of the 1988 SIGCSE Technical Symposium*, Atlanta, GA, February 1988, pages 129—133.

[4] 23rd International Conference on Software Engineering, http://www.csr.uvic.ca/icse2001/.

[5] University of Washington PKU Clinic, "PKU: What is it?," http://depts.washington.edu/pku/whatis.html.

[6] Guide to the Software Engineering Body of Knowledge (SWEBOK), Trail Version, Version 0.9, February 2001, Software Engineering Coordinating Committee (Joint IEEE Computer Society – ACM committee).

[7] Institutionen för Datavetenskap, "Information about Software Engineering Projects," http://www.ida.his.se/ida/kurser/programvaruproj_alla/.

[8] Jeff Offutt, "Software Project Laboratory SWE 626 Course Information," http://ise.gmu.edu/faculty/ofut/classes/626/.

[9] Institute for Software Research, "Software Engineering Education Research," http://www.isr.uci.edu/research-education.html.

[10] Miller,L., Mirsky, S., and J. Huffman Hayes. NUREG/CR-6316, Guidelines for the Verification and Validation of Expert System Software and Conventional Software, U.S. Nuclear Regulatory Commission and Electric Power Research Institute, March 1995.

[11] Offutt, J. and J. Huffman Hayes. "A Semantic Model of Program Faults," published in *The Proceedings of the International Symposium on Software Testing and Analysis*, ACM, San Diego, California, January 1996, pages 195-200.

[12] Philip Greenspun , "Project Schedule for 6.916," http://philip.greenspun.com/teaching/project-schedule.

[13] Pressman, R. *Software Engineering: A Practitioner's Approach, 5th edition*. McGraw-Hill Publishing, NY,NY, 2001.

[14] Shah, Ankur, Sosnowski, Janusz, Van Katwijk, Jan, and Janusz Zalewski. Web-based Course on Software Quality Assurance: Perspectives on Intercontinental Learning. In *Proceedings of the International Conference on Engineering Education '99*, Technical University of Ostrava, Czech Technical University in Prague, August 13 - 14, 1999.
[15] University of Twente, "TGS Core module: Total Quality for Software Engineering: Schedule," http://wwwtgs.cs.utwente.nl/Docs/education/core/soft-engineering/schedule.html.