A TEMPLATE FOR MANAGING AND EVALUATING THE TEAM-ORIENTED SOFTWARE ENGINEERING PROJECT IN COMPUTER SCIENCE

D. V. Pigford
Department of Computer Science
Western Kentucky University
Bowling Green KY 42101
darleen.pigford@wku.edu

Greg Baur
Department of Computer Science
Western Kentucky University
Bowling Green KY 42101
greg.baur@wku.edu

ABSTRACT
This paper focuses on the pedagogical issues in monitoring, documenting, and evaluating team-oriented software engineering projects in a first course in software engineering in a computer science program. Instead of evaluating the finished project at the end of the semester/quarter, the documentation is implemented as staggered deliverables during the software development. Evaluation techniques are exercised during four spiraling and overlapping phases: Project Plan, Ongoing Development, Ongoing Documentation, and Final Project Evaluation. The suggested organization can be a template or basic outline for incorporating the software team experience in a computer science class.

Key Words: Software Engineering, Student Project, Team Project, Team Assessment

OVERVIEW
Integrating successful team-oriented software engineering projects in a one semester/quarter course requires a high overhead for both professor and student time, effort, expertise, and knowledge [1], [2], [3]. Learning to work in student teams, absorbing the complexities of realistic software development, and visualizing multiuser software systems are new and difficult tasks for the best of today's talented computer science students. In addition, the software engineering professor has the added
complexities of evaluating inter-dependent tasks, managing student schedules, negotiating student conflicts, and assigning fair individual grades at the end of the cooperative software development effort [4]. For these reasons both efficient and efficacious techniques are essential in order to enhance student progress and achievement in mastering both the theoretical foundations and applied concepts in a first software engineering course.

These goals can be (partially) implemented by a template for guiding and monitoring student activities in a team environment for software development in software engineering. A structured four phase methodology [1] of

- Initial Planning
- Ongoing Development
- Ongoing Documentation
- Final Project Presentation

can be used to integrate initial learning with a team project.

The Initial Planning involves the formation of teams/customer, setup of communication controls, software project selection, project classification, and project estimates. The Ongoing Development stage is the learning/developing associated a software engineering paradigm. This selected methodology can range from an object oriented model to some variation of a traditional define, analyze, design, code, test, install, and maintain protocols. These concepts are applied to the student software engineering project. The parallel phase of Ongoing Documentation consists of the staggered deliverables associated with the fore mentioned protocols. These student developmental ideas and related documents are then refined into a final oral presentation and organized into a comprehensive project notebook, the Final Project Presentation phase. Hence, the software engineering sequence is divided into small steps appropriate for the academic environment [5], [6].

The body of this paper documents the details of this four phase implementation plan, and the evaluative techniques for the related team project in software engineering.

A FOUR PHASE DEVELOPMENT SCHEME

This section details the four stages of the software engineering project development [7], [8]. This team development of a software system parallels the presentation of the same topics in the regular discussion/example/lecture platform. The four stages are

- Initial Plan
- Ongoing Development
- Ongoing Documentation
- Final Project Presentation

Steps two and three run in a spiraling parallel mode culminating in the final stage of the team software project.

For the Initial Plan, decisions are made concerning the team composition, project (system) selection, customer, project classification, team meeting logistics, examination
of previous student project work, and project estimates. Team composition is determined by the professor. Normally a team consists of four students, sometimes three. Information - address, class schedule, E-Mail, career objectives, part-time jobs, etc - is collected from each student. The professor uses this information along with initial test performance and observed classroom participation as the basis for determining team groupings. An effort is made to balance male/female team composition but not to pair student members who commute long distances. Even with these guidelines, the team assignments still remain "professor judgement."

Each team is assigned a team leader and a type of software system to develop - ACM Member database, Select a College Roommate, Undergraduate Degree Plan Manager, Managing a Video Store, Computerized Drill for Learning Unix, etc. Another interesting alternative plan is to give each team a list of three or four projects to choose from. In addition, a "customer" is associated with each project. The customer is normally a person on campus (or readily accessible off-campus) who is knowledgeable or responsible for the given application. Meetings with the customer are required. After an initial meeting with the team's customer, the team members (as part of a lab assignment) classify the type, environmental, and user interface attributes of the proposed software application by selecting from a professor generated list. See Figure 1. For example, the project, Selecting a Roommate, could be described as a business, informational and report generating system implemented on a stand alone mainframe with keyboard entry whereas the Undergraduate Student Degree Plan Manager is more appropriately a business front-end to a mainframe database with keyboard entry. This classification helps the team write the beginning Narrative that describes the system. A complete project Name --and a project number-- are determined.

Another task of the Initial Plan is the logistics of in-class and out-of-class team meetings. In-class meetings are scheduled ahead of time by the professor. Several ten minute meeting times and one entire class period are scheduled for the semester. All in-class meetings are prescheduled requiring student reactions to specific tasks and or deliverables. One early short in-class team meeting requires the group to compile the

<table>
<thead>
<tr>
<th>Type Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business, scientific, instructional</td>
</tr>
<tr>
<td>Internet, intranet</td>
</tr>
<tr>
<td>Informational, report generating, database</td>
</tr>
<tr>
<td>Front-end, back-end to another system</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environmental Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand alone, multiuser</td>
</tr>
<tr>
<td>Mainframe, microcomputer platform</td>
</tr>
<tr>
<td>Networked, client/server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Interface Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard entry, mouse, etc.</td>
</tr>
<tr>
<td>Text, graphics, audio, multimedia</td>
</tr>
</tbody>
</table>

Figure 1 Student Classification of Attributes for Software Project

following information:
Home phone, work phone
Work/class schedule
Common meeting time
Meeting location
E-mail addresses

Students are reminded to conduct team meetings with (1) an agenda/tasks, (2) start/end times and (3) with an attitude of compromise. Log files are required for each in-class or out of class team meeting. During an early team meeting, the team examines a documentation notebook (last phase) of a software engineering project completed by students in previous semesters.

A final task for the initial planning phase is the determining of project estimates. After students learn metrics like LOC, function points, feature points, and the COCOMO models, they are asked to make estimates of the size, complexity, time, and cost of their software project. Students use person-hours instead of person-months as units in their estimates.

Thus, the Initial Planning Phase is completed when the professor makes team and software project assignments and guides the students to establish communication protocols for in-class/out-of-class meetings, to classify their application, and to produce "rough" student-generated project estimates.

ONGOING DEVELOPMENT AND DOCUMENTATION

The second and third steps of this structured template for software engineering are concurrent development and documentation. As the students "cycles" through class assignments, deliverables such as a Data Flow Diagram, Entity-Relationship Diagram, architectural flow diagrams, scenario, architectural design, procedural designs, screen designs and documentation "shells" are required. Screen designs are connected Visual Basic forms showing the screen layout. A documentation "shell" for a program/module is the internal program documentation of forms/procedures without the usual code. During the coding phase, the actual code is added to the documentation shells. The central point is that deliverables are evaluated, graded, and improvements cited. Students have to be reminded to make the improvements or suggestions given in these early stages of development/assessment. These same/improved deliverables are accumulated for the final documentation binder/presentation (lab 6).

Another task of phase two and three is the creation of a task assignment matrix. This array lists the team members across the top and tasks down the side. All members are required to code, to present an "ongoing" report, and to participate in the final project presentation. A generic task assignment appears in Figure 2. This matrix "grows" as students progress through the project.

Tasks (usually forgotten by students on the final task matrix assignment) include proofing documents, maintaining code backups, collating the documentation binder, and sign-offs on milestone events. The matrix is expanded to include estimates and actual hours required to do the task. For written deliverables, the team member's name and CASE tool utilized are required for the document. In addition, for each of the three lab assignments, one team members presents an oral five-ten minute summary (with transparencies of pre-determined deliverables). Hence, phases two and three are the
spiral and concurrent activities that normally involve the basis of analysis, design, and implementation for a small application, but implemented by a team of students and evaluated by the professor [10], [11].

<table>
<thead>
<tr>
<th>Task</th>
<th>Team Member #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Project Classification</td>
<td>X</td>
</tr>
<tr>
<td>Project Estimates</td>
<td>X</td>
</tr>
<tr>
<td>Data Flow Diagram</td>
<td>X X</td>
</tr>
<tr>
<td>Screen Designs</td>
<td>X</td>
</tr>
<tr>
<td>Entity-Relationship</td>
<td>X X</td>
</tr>
<tr>
<td>Customer meeting</td>
<td>X X X X</td>
</tr>
<tr>
<td>Log Files</td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 2 Sample Task Assignment Matrix

FINAL PROJECT PRESENTATION

The last stage of the proposed methodology is the Final Project Presentation. Each team compiles a project notebook of software engineering deliverables and prepares an oral presentation of their software project. This presentation includes a live demonstration of the software developed by the team. Teams are required to use certain deliverables from their notebook (architectural design, for example) during their presentation to the class, customer, and professor. Major points for evaluating the final demonstration/presentation are given in Figure 3.

Introduction
Transitions between member parts
Use of notebook documentation
Use of specified overheads
Well organized software demonstration
Appropriate Time Interval
Software Engineering Vocabulary
Knowledge of Project
Technical Details of the development tool
Appropriate “Credits”
Lessons Learned by team members
Summary/Conclusions

Figure 3 Evaluation Criteria for the Final Presentation
Systems Engineering
Project (Defining) Narrative ........................................... S1

Analysis
Project Requirements/Specification ................................... A1
  Architectural Context Diagram ...................................... A2
  Architectural Flow Diagram ......................................... A3
  Horizontal/Vertical Partitioning ..................................... A4
  Data Flow Diagram .................................................. A5
  State Transition Diagram .......................................... A6
  Entity-Relationship Diagram ....................................... A7

Design
Data Design
  Data Dictionary ..................................................... D1
  File Formats ....................................................... D2

User Interface Design
  Copy of all Screens ............................................... D3

Architectural Design
  Entire System Design ............................................. D4
  Hierarchy of User Screens ........................................ D5
  Cross Reference .................................................. D6

Procedural Design
  Control Logic Significant Modules ............................... D7

Coding
  Copy of all Code by Modules .................................... C1
  Copy of Generated Reports ....................................... C2
  Copy of All Generated Queries .................................. C3

Testing
  Flowgraphs of Three Significant Modules with
    Cyclomatic Complexity Calculations .......................... T1
  Basis Path Testing Plan ......................................... T2
  Scenarios : Test Plan /Results .................................. T3
  Copy of all Data Files .......................................... T4

Installation/Maintenance
  Installation Instructions ......................................... I1
  Sorted File List/Identification .................................. I2
  Copy of Read.me File ............................................ I3
  User Manual ...................................................... I4

Project Management
  Project Task Assignment Matrix ................................ P1
    (Signed by All Team Members)
  Project Network .................................................. P2
  Project Estimates ................................................ P3

Project Summary/Evaluation
  Recommendations/Future Work ................................... E1
  Lessons learned: Each member .................................. E2
  Technical Tips: Using the tools ................................ E3

Appendices
  A: Transparencies ............................................... AP1

Figure 4  Table of Contents/Grading Form For Project Notebook
The other part of the final phase is the documentation binder. Students are given a predetermined Table of Contents for which they must supply the final updated versions of the designated contents. Additional material can be added to the sections or the appendices. The basic categories in the table of contents include systems engineering (S), analysis (A), design (D), coding (C), testing (T), installation/maintenance (I) [9], project management (P), project summary/evaluation (E), and appendices (AP). The complete list is given in Figure 4. The notation (S1, A2, T3, etc) on the left column in Figure 4 represent the tab section in the notebook where the corresponding deliverable is located. For example to locate the Flowgraph of three significant modules turn in the notebook to the tab section labeled T1 for “testing, document 1”.

This same Table of Contents (shown in Figure 4) is used as an evaluation form. Each document is re-evaluated by marking (on the line of dots) a ++++, ++, or + to indicate an A, B, or C grade for the indicated software engineering deliverable. Minus signs and comments are used where appropriate. A final revised grade team grade is given for the overall project. If there are serious defects (or exceptional work) by a certain team member, then the team grade is altered for that team member. This is why the task assignment matrix and names on each deliverable are so important in determining an individual’s final grade. These project notebooks, minus the evaluative comments/grades, are made available to the next software engineering class as a guide to project planning and software deliverables. The final notebook serves as a guide and learning tool for the next class.

CONCLUSIONS AND SUMMARY

This four phase methodology has evolved over ten years of using teams in software engineering and database classes in a computer science curriculum. This template procedure works well as long as both the students and professor stay on "task". The students usually enjoy a first experience in working in teams, the interaction, and the production of a system which could be used. In general, they are "proud" of their accomplishment, literate on the fundamentals of software engineering, and have some "notion" of the complexity of a "real" system.

This methodology, however, does not solve all pedagogical obstacles of team project in the academic environment. Dominant team leaders, scheduling problems, and the interdependence of individual student work still remain problems for team members. The professor still faces student personality conflicts, accuracy of evaluating individual work, and the dual dilemma of teaching fundamentals along with team management in software development. If academic institutions and industry value the team experience in the education of future computer scientists, then the described methodology is worthy of continued use and refinement.

In summary, this paper describes a four phase pedagogy for integrating a team project in a first course in software engineering for computer science. The four phases include the Initial Plan, Ongoing Development, Ongoing Documentation, and the Final Project Presentation. These steps involve team organization, planning, and
implementing the complete software life cycle for a small application in an academic setting.

REFERENCES