# SOFTWARE ENGINEERING BASED ON THE TEAM SOFTWARE PROCESS WITH A REAL WORLD PROJECT*

*Nasser Tadayon*
*Department of Computer and Software Engineering*
*College of Engineering*
*Embry Riddle Aeronautical University*
*600 S. Clyde Morris Blvd, Daytona Beach FL 32114-3900*
*tadayonn@erau.edu*

**ABSTRACT**

It can be observed the increasing demand for experienced professionals in development of software. On the other hand we see the decrease of interest in students and decline in student's enrollment in computer science and software engineering in most of the colleges and universities in the United States. The outburst of technology in our society demands software engineers to handle the exponential growth. Software plays very important, if not a critical, role in our daily life. It is one of the most rapidly growing fields in Engineering and technology. As software engineering educators, we bear the obligation of attracting and maintaining the interest of students by exploring and identifying initiatives that excite and draw them to the discipline. One tactical move for undergraduate students to understand the concepts and see themselves in their future role is having a realistic customer with real expectations in a software engineering course. There have been several papers exploring the same idea [1, 2, 6]. In this paper, the instructor describes the process and the structure in building a semi-realistic term project, using an external customer for a sophomore level introductory course in software engineering. The course is designed for relatively small groups of traditional and nontraditional students. The instructor assert his efforts in incorporating teamwork with diverse students as well as the challenging experiences that the students faced while working in the semi-real world industrial environment.

**Keywords:** TSP, software engineering, curriculum, real world project.

---

## INTRODUCTION

In general, software engineering is a collaborative process where teams of skilled developers work towards a common goal. The generic goal of software engineering is: "to produce a quality product on time and within budget". The introductory course in software engineering has been identified as one of the core courses for ABET accreditation because of the wide range of practical topics covered in this course. As defined by the Committee for the Computing Curricula Software Engineering (CCSE) in 2001the set of outcomes for an undergraduate curriculum in software engineering should reflect the following: [9]

Graduates of an undergraduate SE program must be able to:

1. Work as part of a team to develop and deliver executable artifacts.
2. Understand the process of determining client needs and translating them to software requirements.
3. Reconcile conflicting objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.
4. Design appropriate solutions in one or more application domains using engineering approaches that integrate ethical, social, legal, and economic concerns.
5. Understand and be able to apply current theories, models, and techniques that provide a basis for software design, development, implementation and verification.
6. Negotiate, work effectively, provide leadership where necessary, and communicate well with stakeholders in a typical software development environment.
7. Learn new models, techniques, and technologies as they emerge.

These abilities in graduates are achieved through a sequence of courses in software engineering. The initial course in this sequence is the introductory course where students get exposed to the concepts. Through practical experiments, students get the breadth of knowledge needed to meet the outcome of the program.

Introduction to Software Engineering is a required course for Computer Science, Software Engineering, and Computer Engineering undergraduate programs at our institution. The students usually take this course in their sophomore year after going through two semesters of programming courses. The syllabus for this course was developed based on students' knowledge on programming. However, there is no pre-requisite knowledge in database or data structure, which makes it challenging. Introduction of a real world customer with their expectation as a term project in the course adds to this challenge. Teaching the introductory course in software engineering, with these challenges was one of the instructor's pleasant experiences.

The students in this course had their first experience in a team work. They saw themselves beyond educational boundaries by considering themselves in a semi-industrial environment and working towards their personal and team goals. It is important and essential to complement the theory of software engineering with practice so that the students see first-hand the importance of the theories and practicality of them. In the instructor's personal experience, the realistic part of software engineering is usually driven by projects or assignments, but the core part of it is the term project. The term project usually starts after one week into the course and continues through the remainder of the semester. Students practice structured development of a product and produce

standard/semi-standard artifacts with respect to their work. They work in relatively small teams with specific roles and responsibilities in order to develop and sell their product to the customer. They also can associate issues like maintainability, quality, cost, and many others, throughout their project work.

It was suggested by the ACM computing curriculum in 1991 that every computer science programs to have at least one software engineering course [9]. Software engineering is the discipline of producing software or the set of tasks that involve development, operation, and maintenance of software. The *ERAU Undergraduate Catalog* describes the first course in software engineering as the student's introduction to the principles of software engineering through participation in a large-scale software development project. The course is designed such that the students learn the principles of development through following a specific process known as Team Software Process (TSPi) as its base process. TSPi was developed by Watts Humphrey and the Software Engineering Institute (SEI) at Carnegie Mellon. Additional processes and procedures augment this base process; however students were restricted to use TSPi process throughout the implementation of the course project.

The exposure to software development phases and defining the set of tasks in each phase needs to be incremental. Students need to get familiar with development phases at a personal level before they can work as a group member. This notion is practiced throughout their initial programming courses as required. Students start by following a well-known and practiced process known as PSP (Personal Software Process) embedded as part of their two semester programming courses. By following PSP, they get familiar with collecting and analyzing their data corresponding to time and defects in their assignments. A detailed coverage of the use of PSP in the program curriculum is given by Hilburn and Towhidnejad in the paper entitled "Integrating Personal Software Process (PSP) Across the Undergraduate Curriculum" [6]. The following sections will explain the TSPi process used in one semester developing the introductory course in software engineering with some activities done in the course. This is followed by the conclusion and some personal observations from the course.

## USING TEAM SOFTWARE PROCESS

The success of software engineers that are proficient in producing software-intensive systems depends not only on their skill but also on their ability to follow well-managed software development processes. Employing and following a disciplined software method is often difficult. There are many factors that cannot all be controlled which play an important role in the use of appropriate processes. Still, there are many organizations that know what they want, but struggle with how to do it. The Team Software Process (TSP), in conjunction with the Personal Software Process (PSP), was intended to offer both a strategy and a set of tasks for using well defined software processes, at both the individual and team level. They are based on the principle that engineers knowing about the job can create a plan, and track their progress and have a certain commitment for their own work. Project tracking requires having a detailed plan and accurate data that can be collected from engineers. It has been claimed and observed that TSP in addition to PSP encourages engineers to balance their workload and maximize productivity, with focuses on the quality [2].

TSP has two main components: a team-building component that occurs during the launch of a project, and a team-working or management component that extends for the duration of the project. The team-building component is composed of goal setting and role assignment that can extend to the development of team process tasks and a detailed plan. The management component is composed of team communication, team coordination, project tracking, and risk analysis. TSPi represents a slightly scaled-back version of TSP intended for college audiences. The main focus of TSPi is on quantifiable process with well-specified roles for each team member, in addition to an eight-stage cyclic system development life cycle. All roles are described in detail including a script of activities and tasks in addition to responsibilities for each of the eight stages.

TSPi process has been used successfully in the School of Computer Science and Software Engineering at Monash University, Drexel University, University of Alberta, as well as Embry Riddle Aeronautical University and other universities and colleges [1]. There have been several reports and papers delivered on the educational aspect of using this process in academia. It has been considered as the base process for capstone courses and graduate courses taught at many colleges [3]. The heavy process requirement of TSPi has been one of the main criticisms from students taking the course, which has also been reflected by other instructors using TSPi in their experiences. Overall, the practice and use of TSPi has been well received by industry and academia. TSPi is a well formed and defined process that can be used for one-semester course. There have been few educational uses of TSPi on projects with substantial industry involvement. In those cases, a measure of support is provided for the teamwork. An example is a case study on the E-Business System formed by a division of an automotive company, which was used in two courses [2]. In some other cases, the involvement is in the form of presentations on the application domain of the project.

Watts Humphrey wrote textbooks [7, 8] on PSP and TSPi and he recommended not using a real customer for the process. Initially, the instructor had doubts whether the use of a real external customer would help motivate students or discourage learning of the process. The objective was to select an appropriate project that has industrial involvement to demonstrate the importance of having and following a well-defined process. To ensure success of the teams, the instructor had to enforce some restrictions on the expectation of the customer. This made the project semi-realistic but the intention was to give the feeling of a real customer for the class. Overall, the experience of using a semi-realistic customer for the course with a real world project has been positive.

In our introductory course in software engineering, students were learning the activities in the TSPi, but the overhead time spent on so-called "paper work" was significant. Most teams were focused and concentrated on delivering the product with all functional requirements and felt that they were held back by all the data collection needed for TSPi. In particular, the students felt that there is a need to have a tool to keep track of their time and defect log instead of doing it manually. An excel tool was used to help reducing the overhead of process tracking data. Unfortunately, the tool itself became the center of many students' complains. A well-developed tool for tracking data in following the TSPi process is still to be developed so that students can concentrate on analysis of data rather than collecting data.

TSPi textbook suggests that the project should be developed over three cycles but due to the time limitation, two cycles were considered more appropriate. The general set of tasks for each cycle was defined as follows [8]:

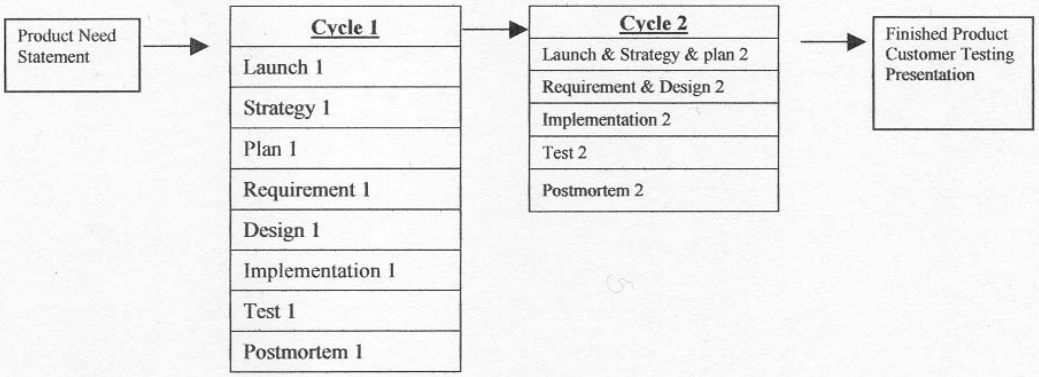


**Fig1** TSPi process structure and flow

## COURSE ACTIVITIES

One of the main objectives of this course is to give students the ability to experience problems in large software system development, discuss issues, principles, methods and technology associated with software engineering theory. They practice tasks of software development (e.g., planning, requirements analysis, design, coding, testing, quality assurance, and configuration management), by working as part of a team. Overall, the students are expected to learn how to use a software development process to develop a high-quality software product in an effective manner. The topics covered in support of the objective in the course include:

- Introduction to Software Engineering
- Personal Software Process
- Models of Software Process
- Project Planning and Organization
- Software Requirements and Specifications
- Software Design Techniques
- Software Quality Assurance
- Software Testing
- Software Tools and Environments

This course is a four credit hour course, designed as one hour of lab and three hours of lectures per week. The usual class size for this course is around 20 students. The fact that the class size is relatively small allows the formation of students into teams of four to five where each student is given specific role in the team. Even though all students entered in this course have some knowledge of software development at a personal level (PSP) a general review of PSP with a programming exercise is needed to ensure their

basic understanding. Each individual student is given a relatively simple programming assignment in which they follow PSP2.2 which has additional tasks of code and design review in compare to PSP. The main activities of the project can be defined as follow:

*1. Forming the team:*

The intention of adopting a team approach in the course project supports and meets several parts of the program outcomes. Before forming the teams, a number of defined roles with specific responsibilities for each are described and studied by students. Students are then asked to rank their preference within the specified set of roles in the team and any preference they may have for their team members. They also indicate their schedule for the semester which identifies their prior commitment and availability. Selection of team members and roles must be done very carefully since it will have a major contribution to the success or failure of the team. The specified roles in the team are defined as:

1. Team Leader
2. Development Manager
3. Planning Manager
4. Quality/Process Manager
5. Support Manager

There is a set of responsibilities associated with each of the roles [7]. The role of quality/process manager and support manager is usually combined and assigned to one student to balance the workload more properly for the project. An effective team leader is necessary, but not sufficient. A competent programmer as development manager is essential. The planning manager and quality/process manager with responsibility of support manager are critical in the overall team success. The most important factor to the success of the team is the ability of the members to work together efficiently.

To reduce the likelihood of problems in the team and build high confidence among team members, the selection of team members must include at least two strong programmers. The students are asked to describe their experience in programming, their interest and any other team or leadership experience that they may have had previously. After collecting all the information required, the instructor forms different teams of four or five students based on their preference, their schedule for the semester, and their transcripts. However, their roles in the team are suggested based upon their preferences and supporting evidence. If they have had any leadership experience and have identified their first choice preference as the team leader, they are suggested to become one. The role of development manager is the one that usually has the least preference in their list, and it is suggested based on the programming knowledge in their transcripts. The planning managers are selected based on preference and the result of the PSP assignment. The quality/process managers usually assume the responsibility of support managers and are selected based solely on preference. The team members have to be assured of available time during the week for their meeting by matching their schedule.

If the whole class cannot fit into teams of four, the task of quality/process manager and support manager are given to individuals so that some teams may have five members. In general, the team will succeed if the selection of students is appropriate and the

progress is monitored on a weekly basis by their deliverables. If there is any delay in deliverables or any complaint by a member of a team, all team members are asked to see the instructor individually. They are then consulted by asking their opinion on other team members and their suggestion for resolving the issue. In most cases, the students overcome the problem by themselves and do not require the instructor to interfere. One major problem in using TSP is that the load is not distributed evenly corresponding to the role, however, everyone in the team is strongly encouraged to help each other since the major grade for the project is assigned as team grade. This way, every one must ensure the deliverables of the project are done on time.

*2. Project Need Statement*

All teams work on a common project that has an initial need statement. The need statement for the project was produced by the collaborative work of the instructor and the external customer. The customer for the project was a manager of a local River Cruise Company, who used forms and papers to manage their reservations. She was keeping track of her employees' records and inventory by using specific forms. The software product was to automate the scheduling and reservations and also keep track of her historical data in order to analyze them at a later stage. At that time, one of our students was working part time for her, helping with the reservations. After the initial conversation with the customer to get a general overview of her need, It was determined that the complexity of required system was appropriate for the course. Although the need statement clearly identified basic functionalities of the required system, it lacked specific details of system usage. The students were given the need statement to develop a set of questions to ask the customer. It was also explained to the customer that the class would have to follow a specific process and produce artifacts like "Software Requirement Specification", "Software Design Specification" and a user manual during the course. It was also pointed out that the business deal for the product, if any, should be outside the class and the university. As far as the class was concerned the customer was a representative of a real customer for the project within the scope of the class and not necessarily a purchaser. After students were given time to read through the need statement and prepared a set of questions the customer was invited to participate during a class session to interact and go over the elicitation. The customer gave an overview of the system requirements by explaining the process that she takes to reserve passengers on the tours. The students had many questions to ask with regard to specific details of system and the customer responded to most of them. The students were asked to record all interaction, and a specific question and answer document were developed as an amendment to the initial need statement. The complexity of the problem seemed appropriate, but the lack of required knowledge in database was a concern. The instructor had to ensure this would not cause a major problem and each group had at least one student who had taken the database course in their previous semesters.

*3. Project launch and strategy*

After forming the groups, during the project launch session, students were asked to write down their team and individual goals in a specific form. The textbook identified an

initial set of goals for team and members [8]. For the first cycle, most teams adapted the same goals but they were asked to identify more reasonable and measurable goals for the second cycle. The students were informed about the format and necessity of holding regular weekly meetings. They were also told how to set up communication among members when they held their first team meeting during the lab hour. Some teams adapted online group to help with the access, communication and configuration. Subsequent to discussing the process, the team was asked to develop a simple rough draft of the conceptual design of the system to be used for the size and time estimation. They were also asked to identify functionalities associated with cycle one and two. During the launch and strategy of the second cycle, they were asked to assess risk issues and identify likelihood and impact of risks in their process.

*4. Project planning*

The cycle one planning phase was, by far, the hardest task for the team. The students had some experience in individual planning using PSP set of tasks, but not in a group setting. They were given a sample plan from previous similar project with specific instructions as how to develop their plan for cycle one. An Excel tool containing several spreadsheets was used for planning and the application of the tool was demonstrated. However, the tool gave many problems when it was not used properly by the students. Losing data in the tool due to simple mistakes was frustrating for them. Once the team identified their tasks and planned individual time for each task, the planning manager enters their actual data on a weekly base. The tool would not allow any member to view or change their planning time after the actual times were inserted. To facilitate teams with their problems, the instructor had individual sessions with planning managers. The second cycle was much less troublesome since most teams knew in advance about the behavior of the tool. Overall, their planning and actual data for the second cycle were closer due to their awareness and experience of the first cycle.

*5. Project Requirement*

This is the first course that requires students to develop the SRS (Software Requirement Specification) document. During the lectures, the importance of requirement specification was mentioned as well as the outline for SRS using the IEEE standard. In depth understanding of requirement phase would require several week from the course schedule. To ensure their steady progress in the project and success on the requirement phase, the students were given a sample SRS as a template to follow. In addition to following the outline of the sample SRS document, they had to add a system prototype in their requirement specification document and develop system test plan. The teams were asked to do a formal review and inspection of their SRS and system test plan as part of their requirement phase activity. The importance of tractability (from need statement and QA documents) and completeness with stress on following a specific format for the SRS and test plan was emphasized. The students were asked to examine the documents individually and during one of the lab hour, a formal inspection would take place. They were guided thought the inspection meeting and produced an inspection report. They also developed and inspected their system test plan during this phase. After removing detected

defects from the SRS document, it was reviewed by the instructor and the grader, but not by the customer before they were signed off and returned to team members as an agreement document. The teams were asked to develop the requirement specification for the whole system and not just the product for cycle one. During cycle two, another review and inspection was done on the SRS to ensure consistency.

*6. Project Design*

Another major problem was lack of knowledge by team members about how to design a system. The students would learn design methodologies in their "Analysis & Design of Software System" course that has the introductory course as prerequisite. In the introductory course the instructor covers a broad aspect of different design methodologies. The students are stepped through a deign process known as SOBAD (Simple Object Based Architectural Design) developed by Hilburn. The SOBAD design process is used by following scripts and filling out specific forms which compose part of SDS (software design specification) document. A sample design specification document with particular format was provided to students. The integration test plan was developed during this phase. The students had a formal review and inspection of their SDS as well as their integration test plan in both cycles. They were all given feedback for their SDS and test plans.

*7. Project Implementation, testing and postmortem*

During the implementation and testing, the development managers took over and did most of the coding, and the quality/process managers were in charge of testing. At this stage the teams were competing against each other. The testing was not followed correctly during the first cycle since teams were trying to add functionalities at the last minute. Teams did not follow their test plans and it looked like an informal and unorganized testing during development was done. The customer tested the cycle one product which gave teams some exposure to customer testing. Each team was asked to demo their product to the customer before being asked to go through specific scenarios. During the cycle one postmortem, teams analyzed the data and identified ways to improve the cycle two process, especially in their planning. Team members also realized the need to spend more time in testing to ensure a better quality of the product. They managed their time much better for second cycle development. All teams except one were well prepared for customer testing of the final product. However, all teams finished most if not all the required functionality of the system. The alpha testing was very successful and impressed the customer. There were few students that spent over forty hours during the last week of the project for coding and testing. Most teams underestimated required time for the project but overall were very pleased with the teamwork practice. The students gave a final presentation, describing team organization and description of work processes (decision making, role assignments, meetings, etc.). They analyzed the process and product by using size, effort, and quality data. In their presentation, individual team members identified what value this project had for them by specifying their success, failures and lessons learnt. They evaluated their project and the course and mentioned their recommendations.

## CONCLUSION

The customer was happy with the result of alpha testing. All teams had nearly all functionalities implemented but with some minor defects but the students felt they should have spent more time on testing. In instructor's experience, the presence of external customer and enforcing the TSPi process worked well and resulted in an environment of competition to get the best product to please the customer. The disadvantage was that the students gave more attention to the product than the process. The students felt that the customer was interested in the product. But the artifacts like SRS and SDS with the test plans and planning data had been too much over head. They had few errors in their data because of misunderstanding, or providing false data due to limited time. The two-cycle development helped to improve their collection of data, and in the second cycle, the planning managers had been more active and aware of planning artifacts and more proactive in collecting data. Overall the introduction of a semi-customer was a rewarding experience in complementing the term project and making it close to a real world project.

## REFERENCES

[1] Martha E. Myers, Charlotte S. Stephens "CONTINUING IMPROVEMENTS IN TEAM PROCESS: SOFTWARE ENGINEERING APPROACHES AND PEER INSTRUCTION INTERVENTIONS" in *Proceedings of the 17th Annual Conference of the International Academy for Information Management*, pp. 148-149.

[2] A. L. Steenkamp, "A STANDARDS-BASED APPROACH TO TEAM-BASED STUDENT PROJECTS IN AN INFORMATION TECHNOLOGY CURRICULUM", Proceedings of the 17th Annual Conference of the International Academy for Information Management. pp. 54-62.

[3] Towhidnejad, M., "INCORPORATING SOFTWARE QUALITY ASSURANCE IN COMPUTER SCIENCE EDUCATION: AN EXPERIMENT", 32nd ASEE/IEEE Frontiers in Education Conference, Boston, MA. November 6-9, 2002.

[4] Anthony Lattanze, Manuel Rosso-Llopart, and James Tomayko "Teaching the Personal Software Process and the Team Software Process in Carnegie Mellon's Master of Software Engineering Program".

[5] Thomas, B. B. and Morrison, B., "Ventures into Capturing Effort in Programming", 2002 ASEE Southeast Section Conference.

[6] Hilburn, T. and Towhidnejad, M., "Integrating Personal Software Process (PSP) Across the Undergraduate Curriculum", Proceedings of 1997 Frontiers in Education Conference, November 1997, pp 162-167.

[7] Humphrey, Watts S., "An Introduction to the Personal Software Process", Addison-Wesley, Reading, MA, 1997.

[8] Humphrey, Watts S., "Introduction to Team Software Process", Addison-Wesley, Reading, MA, 2000.

[9] Committee for Computing Curricula, Software Engineering http://sites.computer.org/ccse/