DISCRETE EVENT ROLE PLAYING SIMULATION OF SMALL

TEAM SOFTWARE ENGINEERING PROJECTS

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include any proprietary or classified information.

_____

Neal L Rogers

Certificate of Approval:

_____	_____
Richard Chapman Associate Professor Computer Science and Software Engineering	David Umphress, Chair Associate Professor Computer Science and Software Engineering
_____	_____
Dean Hendrix Associate Professor Computer Science and Software Engineering	Stephen McFarland Dean Graduate School

DISCRETE EVENT ROLL PLAYING SIMULATION OF SMALL TEAM SOFTWARE ENGINEERING PROJECTS

Neal L Rogers

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 7, 2006

DISCRETE EVENT ROLE PLAYING SIMULATION OF SMALL

TEAM SOFTWARE ENGINEERING PROJECTS

Neal L Rogers

_____

Signature of Author

_____

Date of Graduation

DISSERTATION ABSTRACT

DISCRETE EVENT ROLE PLAYING SIMULATION OF SMALL

TEAM SOFTWARE ENGINEERING PROJECTS

Neal L Rogers

Doctor of Philosophy, August 7, 2006
(M.S. Columbus State University, 1998)
(B.S. Columbus State University, 1972)

57 Typed Pages

Directed by David Umphress

A dissertation presented on the simulation of small team software engineering projects. Each individual acts as a team leader for a software project that is scheduled to last from 5 to 52 weeks. A team of four or five 'software engineers' is simulated by the system. Metrics obtained from the Personal Software Process (PSP) exercises are included with each team member. Next, task assignments for each team member are entered for each weekly time period. The system then simulates each team member's activities and presents the results to the team leader. The team leader can modify assignments for the next period and the system continues to simulate all team activity.

The system simulates possible problems and defects that a team could face in an actual project. The team leader can use the system to practice team leadership skills and build experience in directing a team to complete a project on time and within budget.

The simulator was validated statistically by analyzing simulated results and actual project results.

Style manual: Software Engineering

Software used: Microsoft Word 2003 SP2

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

INTRODUCTION

Although much progress has been made in representing and simulating software development processes, significant work remains. The majority of research to date has been directed in simulating processes from the top down using Systems Dynamics as well as Monte Carlo techniques, thus providing decision makers with information about the strategic effects of processes. With exceptions noted later, relatively little effort has been expended in bottom up, role-playing simulations that provide workers with insight as to how processes work and can be managed.

This research describes the development of a prototype tool for simulating small-team software processes from a role-playing perspective. A user of the simulation tool is placed in the role of lead software developer in a computer-simulated project. Using a game-playing paradigm, a user guides a simulated software project from start to completion. In so doing, a user gains hands-on experience with instantiating, manipulating, and observing software processes.

A user interested in learning about the role of a software engineering team leader provides the simulator with the planned tasks and team member schedule with assistance from the tool. She also decides how many developers to include in the simulated project together with their performance data, such as productivity, defect rates, etc. Additionally, she provides information on the scope and size of the software to be "built" during the

simulation. The project is simulated based on values randomly generated along distributions defined by the programmer performance data. Planned versus actual metrics along with results of developers are displayed to the user. The user has the opportunity to alter resources, modify tasks, or make other changes to the project and observe the consequences of those changes.

The research was validated by comparing the simulation results to actual project data. The Team Support Process (TSP) [Humphrey 2000] was used as the base for the model. TSP offers several advantages as a concrete process model to use for the purposes of validation. First it was designed for small teams such as might be found in an industrial training or educational environment. Second, it is explicitly defined. Third, it is in use in industry and is, therefore, not strictly academic. Fourth, it has both a managerial and a technical component: TSP is the umbrella process that orchestrates technical subprocesses articulated through the Personal Software Process [Humphrey 1995], or PSP. This approach allows a user to assume a management role by working at the TSP level. Finally, lessons learned from using TSP in industry can be incorporated as heuristics in providing assistance to the user of the simulation.

LITERATURE REVIEW

Existing work falls into two major areas: simulation and software processes. Below is a synopsis of the present state of knowledge in each area. This work will concentrate on identifying the process elements required for a realistic simulation.

Simulation

The major roots of simulation specific to software projects began in the late 1970s with Thayer [Thayer 1979] providing a static model of managerial processes common to software projects. While the model was qualitative and not designed to be exercised dynamically in a simulation, it provided a critical glimpse into identifying activities that take place during any software development project. McFarlan [McFarlan 1974] provided a complimentary model by explaining the degree to which managerial processes affect the software development effort. His model was based on project characteristics such as size, complexity, and formality. McKeen's research [McKeen 1981] provided increasing specificity by modeling and examining the managerial and technical differences among lifecycle activities within a project. As with Thayer, McFarlan and McKeen were concerned with static models that identified process activities. Their efforts resulted in models that were not detailed enough to simulate a project directly; however, they provided foundational research for subsequent behavioral models.

Abdel-Hamid and Madnick [Abdel-Hamid et al 1991] produced the first recognized approach to simulation of software management processes. Their work is considered seminal to the process simulation community. They identified processes to simulate from Thayer, McFarland, and McKeen, as well as by their own research. They expressed the behavioral aspect of their processes using the complex system dynamic approach developed by Forester [Forester 1961]. The result was software that, when executed, mimicked the human resource management, controlling, planning, and software production facets of a software development project during design, coding, and testing activities. From this they were able to draw general conclusions about the influence of software estimation, the economics of quality assurance actions, and the validity of previously anecdotal staffing wisdom. Since the 1991 milestone, Arizona State University (e.g., [Merrill and Collofello 1997]) has been active in expanding the Abdel-Hamid and Madnick's work to include a chapter in the book: "*Simulation-based Software Process Modeling and Evaluation*". [Ros et al 2004] Other relevant contributions include those of Schneider [Deininger and Schneider 1994], who reports on using general simulation models to teach software project management, and Tausworthe and Lyu [Tausworthe and Lyu 1996], who predict the reliability of a software product based on the quantification of software project characteristics.

Abdel-Hamid and Madnick's work remains the most comprehensive to date. Indeed, emerging from it has been a significant process simulation community. Process simulation workshops (e.g., ProSim '98 through ProSim 2005) have yielded significant work. Emergent themes within the community include simulating processes to facilitate strategic decision making, to improve performance, examine return on investment, and

developing rapidly deployable software process models. [Raffo 2003]. The Portland State University group has been especially prodigious in applying both system dynamics methods as well as discrete-event hybrids in mining simulated processes as much as possible for bottlenecks, resource tradeoffs, and quality assurance flaws. Their work, together with the Software Engineering Institute's examination of process modeling (e.g., [Burke 1997] and [Christie 1999]), has subsumed the most relevant work of the large software process conferences, such as the International Software Process Workshop, the International Conference on Software Process, and the International Conference on Software Engineering.

The bulk of the process simulation efforts have been directed toward top-down repetitive execution of processes. The simulation technique of choice is that developed by Forester's System Dynamics method. This technique allows for a "big picture" view of a system by defining information flow, flow rates, and causal loops. The technique is also implemented in a number of languages, including the highly graphical iThink system. The disadvantage of the technique is in its coarse granularity. System dynamics models represent systems as a collection of activities with interconnecting flows; the model is simulated by flowing information between activities in much the same way as water flowing from one container to another over an interconnecting pipe. Mechanisms exist to restrict the flow of information; measurement probes reveal to the user of the simulation the rate of flow and thereby give an insight into the operation of the system. Similar activities are generally represented as abstract collections. For instance, in software development project simulation, the programmers would be considered as a group of anonymous entities possessing a single statistical characteristic expressing, say, code

productivity. Simulations using this technique are, by nature, designed to reflect broad trends and overall statistical performance.

The complement to system dynamics is discrete-event simulation (see [Robinson 2005]). In contrast to the analog nature of system dynamics, discrete-event offers control over distinct elements within a system. In other words, rather than viewing the model from the flow within the system, the model can be viewed from the perspective of each individual entity being simulated. Such an approach could be used to model configuration items (such as requirements, code modules, bug reports, documentation, etc.) more intuitively for training purposes. Moreover, the researchers believe the discrete-event approach can more accurately reflect a specific development project's circumstances (e.g., Sally has a particular code defect rate, Joe has another - each measure can be tailored to the individual's real-life characteristics).

While both system dynamics and discrete-event simulations have been used for instructional purposes (see [Martin and Raffo 2000]), the number of times a process has to be performed to obtain statistically valid results necessitates a top-down perspective, one that discourages interaction. These methods are excellent for analyzing and expressing theoretical flow of information through a system and assume that all flows are carried out in the real system; however, in actual development efforts, processes are circumvented and activities are omitted. The University of Stuttgart's Institute of Computer Science is at the forefront of instructional use of process modeling by incorporating interaction into process simulation through role playing (see [Drappa and Ludewig 2000]). Their system, SESAM, relies on a "player" to intercede as the model is being exercised from one discrete time unit to another. While limited to a text-based

interface, SESAM provides a modeling paradigm based on rules and activity as well as a model executor that generates performance statistics of virtual software developers.

Other instructional offerings include “Problems and Programmers” an educational software engineering card game [Baker et al 2003] where two teams attempt to complete a project with limited time and resources. Various problems occur when your opponent plays cards that can restrict your options or remove previously completed work.

SimSE is a graphical simulation [Navarro 2005] that shows a team of engineers, each with various levels of proficiency at gathering requirements, design, coding and testing. Some have personal quirks, like ‘not working well with others’, or ‘only enjoys testing’. The user assigns tasks to engineers and can also set the time limit to stop and review assignments. Several options to view what each engineer has accomplished, number of discovered defects, how much of the resources have been consumed, etc. are available to the user. A percentage ‘score’ is issued at the completion of the simulation.

Software Processes

The principle that software could be built in a deliberate fashion was first formalized by Royce [Royce 1970], where he identified all projects as having a lifecycle, or collection of common technical software development activities. Throughout the next two decades, numerous lifecycle processes were proposed, such as the spiral model [Boehm 1988], the object-oriented life model [Henderson-Sellers 1990], and the synchronize-and-stabilize model [Cusamano and Selby 1997]. The strength of all of these approaches is that they advocate a disciplined approach to the technical aspects of building software. Their weakness is that they do not adequately address managerial

functions nor do they speak to the ancillary obligations of configuration management, quality assurance, training, and similar issues.

The realization that software development consists of more than technical activities came as a result of the lifecycle model research as well as the manufacturing process improvement work of Deming [Deming 1986], Juran [Juran 1988], and Crosby [Crosby 1979]. The marriage of these areas resulted in published process standards that advocate using a combination of technology, management, and process to build software. IEEE 1074 [IEEE 1997] is one such an example of a published process standard. Because the standards are *prescriptive* in nature, projects within the same organization using different process standards cannot readily benefit from each other's personnel, training, and lessons learned. This led to development of *descriptive* process models, i.e., models that describe desirable technical and managerial activities but do not prescribe when or how those activities take place. The ISO 9000 series [ISO 2000] and the Capability Maturity Model Integration [CMMI 2002] are two such descriptive models that help organizations assess their ability to build software based on the quality of their defined software development processes.

The process models advocated throughout the 1990's have led to a recent backlash among smaller development organizations. Laitinen [Laitinen et al 2000] suggests that the software developers, feeling encumbered by the bureaucratic requirements of software processes, are seeking to trim down processes to manageable levels. Fowler [Fowler 2000] concurs by noting that software firms are adopting so-called lightweight models that adapt to their particular circumstances. Lightweight process models such as Extreme Programming [Beck 2000], Crystal [Cockburn 2000], Feature Driven

Development [Coad 1999], and Scrum [Schwaber 2000] reflect this cultural evolution by focusing on adaptation rather than prediction, which is the focus of heavyweight models.

Lightweight models rely on rules and philosophies to impart process to their adherents. This makes them particularly difficult to simulate because of the lack of structure in the processes. Humphrey's umbrella managerial Team Software Process (TSP) [Humphrey 2000] and technical Personal Software Process (PSP) [Humphrey 1995] and PSP, A Self-Improvement Process for Software Engineers [Humphrey 2005], on the other hand, provide a notable exception. TSP and PSP have structural process scripts that describe explicitly at a particular level of abstraction what actions participants of the process should be doing, thus making them suitable for simulation. Moreover, because TSP and PSP are used in both academic and industrial settings, they have empirical usage data. (see also [Humphrey 1998], [Hilburn 1999], Borstler et al 2002], and [Umphress et al 2002])

RESEARCH DESCRIPTION

Software engineering students need a method to practice their newly learned skills in project management. The method should be one that is current and is actually used in both academia and industry. PSP [Humphrey 1995] and [Humphrey 2005] and TSP [Humphrey 2000] are processes for personal and team use respectively and are used for instruction and also used regularly in industry. [Umphress et al 1995]. PSP, or Personal Software Process, is intended for individual use and will result in determining personal metrics such as: lines of code produced per hour, errors per thousand lines of code, percent of time spent in each phase of the life cycle, etc. Some of these metrics are used in simulating the activities of the team members. The TSP, or Team Support Process, was designed for small teams and is the process that is used to coordinate the activities of the team.

The primary objectives of this research were:

- To ascertain the type and specificity of knowledge about small-team software processes required to conduct meaningful and valid simulations. Although TSP will be the specific process model used, information that can be generalized to other process models will be produced from this research.
- To develop a means by which to exhibit process interaction and performance. Predominant process simulations systems to date interact with the user by

displaying a symbolic portrayal of the system being simulated accompanied by widgets such as meters or graphs that portray process metrics [Abdel-Hamid and Madnick 1991]. Text-based interfaces seem to prevail for role-based simulations [Drappa and Ludewig 2000]. This research takes a different approach and portrays the process as it is simulated by animating the virtual people involved in the process. This approach, similar to Maxis *The Sims*, gives a novel perspective on process learning; and, although we propose to only prototype the interface, we feel the approach will lend new insight into instructional tools.

The ultimate aim of this research was to produce a software tool that simulates a software development project. The tool was validated by comparing actual results from a project completed by several teams with an identical number of simulated results. This work offers significant contributions to the current state of software process research in the following ways:

The simulator itself will serve as a means by which software developers can learn about processes, specifically the Team Support Process. The TSP assigns roles to the team members. The roles consist of: Team Leader, Development Manager, Planning Manager, Quality/Process Manager, and Support Manager. Each role has specific functions to perform for the duration of the project. Forms to assist with the planning, scheduling, and tracking of the project are provided with TSP. A project workbook (The TSP tool) is also provided. It is an Excel workbook with sheets that contain many of the forms needed and is designed to minimize the manual effort required for planning and tracking the project. Use of the simulator will give students valuable practice using and

understanding the TSP tool and the underlying processes. In particular, the tool will help users understand what information must be supplied to a process to make it successful. For example, they will learn that if they do not assign enough time for review and inspections, numerous defects will be left in their product. The tool will also allow users to conduct 'what if' scenarios regarding process issues. Finally, by using PSP and TSP, tool users will be engaged in simulating processes using *their* actual performance data, not data that was manufactured for simulation purposes.

Using the Team Support Process as a baseline allows us to explore in a simulated environment the shortfalls and assumptions of the process and apply those lessons to the industrial practice. TSP, and its more-detailed companion PSP, would be validated as being suitable for simulation.

Many approaches to simulation of team results have been implemented, but not many have attempted to simulate each individual and then assemble the results for the entire team. This work simulates each individual using his/her production metrics obtained from the PSP process, and combines them to determine the team results.

A major problem with teaching software engineering is that most students never get a chance to see what could happen as a result of different choices made during a project. This work allows a student to act in the role of a project manager and get experience in making decisions and seeing the results rapidly. Since TSP is used as a basis for this work, a student who has taken (or currently taking) a TSP course will have the information required to execute the simulation. The current simulator implementation consists of several Excel worksheets added to the TSP workbook, which the PSP/TSP students will be familiar with as they are used in both PSP and TSP instruction.

Four metrics are entered for each engineer: Lines of Code (LOC) per hour, Errors per Thousand LOC, Pages per hour, and Errors per page. These metrics are entered with each engineer on the Candidates sheet and are transferred to the task sheet to assist with the task planning. The metrics were chosen because they are results of the PSP course 'final report'. [Humphrey 2005] TSP assists the user in developing a task list which consists of the tasks needed to complete the project and the planned hours for each engineer for that task. The metrics from PSP help determine the plan time for each engineer's tasks. Once the plan tasks are entered, the team hours planned per week are entered into the schedule.

The model is ready to execute when the team is selected, roles are assigned, and the plan (tasks and times per engineer), and weekly schedule have been entered. Once the team leader determines that the plan is how she wants it, she will go to the Simulator sheet and click 'Simulate next week activities'. The simulator will then simulate tasks for each engineer in the sequence presented.

A random function is used for most of the 'actual' calculations. It is designed to allow the instructor to 'throttle' the allowable range of the results. The Ranges sheet has several categories of activities that the instructor can enter a minimum and maximum percent value with 100% resulting in the normal value. For example, if the instructor entered 50% as the minimum and 150% as the maximum, an activity could range from 50% below to 50% above its nominal range. For each task for the current week (and including the next week, if there is extra time) the simulator will process the following pseudocode until each engineer's time is expended, or the project is completed.

```
'  Executed for all tasks (thru current week number + 1) until each engr time used

'   Simulate 'actual' hours to complete each task for the simulated week
‘   i  is the task index, e is the engineer index
    Do While ‘incomplete tasks are left’
       If  Task(i) ‘not completed’
          e = findEngr(i)                    ' Set e to index of current engr
          taskHrs = taskTime(i, Week)     ' Get time spent on task this week
          If taskHrs >= 0.1 Then             ' Only if task worked on this week
             Post taskHrs to the task and reduce engr hours left
             Post to Actual Metrics for engr 'e'
             Accum metrics
             Accum defects
          End If
       End If
       i = i + 1
    Loop
```

Basically, the next task is selected, the engineer is determined, and the plan time is multiplied by a random factor to determine the actual time expended on the task. If the engineer has enough time left for the current week, the task is completed, otherwise the time is posted to the task, but the task is not marked as completed. The time is charged to the engineer and posted to the phase of the task. Last, the defects introduced (or removed) are determined and posted by executing the following pseudocode task:

```
    Select Case Range(task phase)                              ‘ Select current task

        Case ‘Phases that usually adds to defects – Documentation type’
          If (Size measure = "Page") Then                  ‘Check for Page size
             defAdd = engr error per page * number of pages _
               * Randy(minRange, maxRange)
          Else                                                    ‘not Page, use Time
            defAdd = engr pages per hour * engr errors per page _
              * actual hours * Randy(minRange, maxRange)
          End If
```

```
        Case ‘Phases that usually adds to defects – Coding type’
           If (Size measure = "LOC") Then                    ‘Check for LO’ size
              defAdd = engr err/KLOC * LOC / 1000 * actual time _
                * Randy(minRange, maxRange)
           Else                                              ‘not LOC, use Time’
             defAdd = engr LOC/hr / 1000 * engr err/KLOC _
              * actual time * Randy(minRange, maxRange)
           End If

        Case ‘Phases that usually reduces defects’
           defRem = Task actual hours * Errors corrected per hour _
              * Randy(minRange, maxRange))

        Case ‘Phases that usually has no effect on defects’
          No effect on defects
    End Select

    ‘post defects to all defect tracking areas
    defPost = defPhasePost(defPhase, defNdx, defAdd, defRem, i)
```

The actual results are posted to the simulator sheet showing the time spent by each engineer and any defects introduced or removed with a running total of defects left. Several summaries are presented on the simulator sheet that the project manager can analyze and use to make changes to the plan. First, there is the ‘team member planned phase totals’ that lists each engineers planned time broken into seventeen categories defined by the TSP. This summary is available before any simulation is completed so the project manager can adjust tasks between team members and phases until she is satisfied with them. Once the simulation is started, five other summaries are presented to the team leader. The ‘actual phase totals’ show the time spent in each of seventeen phases along with the percent of total time. The ‘project summary’ shows weekly defects added and removed with a total at the top. The ‘team ratio’ summary shows the goal, plan and actual times for three ratios defined by TSP: Design as a percent of code, code review as a percent of code, and design review as a percent of design. TSP suggests 100%, 50% and 50%

respectively as goals. The ‘defect by phase’ shows defects injected, removed, and remaining, broken into five phases: planning/requirements, high level design, detail level design, code and test. The ‘plan and actual time in phase’ shows the time and percent of total time in six phases: Management and analysis, planning, requirements, design, implementation and testing.

Simulation continues until all tasks are completed, or there is no more time left. If the instructor allows, additional weeks can be added to the schedule if needed so the project can be completed.

There are also several graphs on the project sheet that can help the student visualize exactly what is happening and why. The Quality Profile shows how closely the plan follows the quality guidelines by showing a pentagon with five ratios with values between zero and one whose products give the process quality index. [Humphrey 2005 – pp150-153] A few other charts that are very helpful in visualizing the progress are: Earned value per Week, Cumulative Plan vs. Actual Hours, and Cumulative Earned Value. The charts can be viewed at anytime during the projects duration.

APPLIED RESULTS

To use the simulator, several pieces of information need to be entered. First the instructor enters a list of team member candidates that include certain metrics that were obtained from their PSP assignments. Then the instructor enters the ranges that the team members may deviate from their normal production results. Watts Humphrey recommends that about 50% of the student's PSP production results should be used for most projects. The instructor can use her own experience in determining the factors. Most of these ranges are expressed in percentages from the expected results. There are also a few ranges expressed in units per hour, and probability of completion. The instructor could also set a limit for the defects remaining after a project is completed, and have the student make another run using the information learned. Each team member would then gain insight into the decisions required of a team leader / manager and learn management skills during the process. The instructor can also enter error messages that would be unique to the particular project.

The student, or candidate team leader, enters the project information on the Project sheet. The project name, team name, start date, instructor, and cycle are entered and the 'Use Defect Log' is checked. The 'Use Time Log' is unchecked. She then selects a team of five engineers from the Candidates sheet and assigns each a standard TSPi role on the Roles sheet. The standard TSPi 'SUMS' worksheet data is then entered. This usually

consists of entries for the System Requirements Specifications (SRS), High Level Design (HLD), and major components of the system. These entries include estimates of size, usually either in Pages or Lines of Code (LOC). It is highly recommended that only major components be entered on the SUMS sheet because each entry will generate several detail lines on the Task sheet.

She then clicks on the 'Generate Task List' button on the Task sheet. Several tasks will be generated for each major component listed on the SUMS sheet. Estimates of time, estimated size, and size measure (Page or LOC) are entered next for each engineer on each task. The tasks should be rearranged into the order they should be completed, if necessary, and the workload should be balanced between the team members. If multiple team members are to work on a given task, enter the time for each in the appropriate Role column, and the total estimated size. A future step will create individual tasks and divide the estimated size equally between team members. If the size should not be distributed evenly between the team members, a separate line can be entered for each member, or the size can be adjusted for each engineer.

The planned hours for the entire team for each week are now entered on the Schedule sheet. The times for each week do not need to be the same, but it is usually easier if they are initially. Now return to the Task sheet and click on the 'Split Plan Time' button. This selection will generate individual tasks for each engineer, and generate the proposed schedule. She will get an error message if there was not enough time entered on the Schedule sheet. If an error is shown, return to the Schedule sheet and add more time. She can extend the project by a week or more if that is an option, or simply add more time for each week. If any entries on the Task or Schedule sheet are changed or moved, the 'Split

Plan Time' button must be clicked again. The 'Split Plan Time' function also updates the planned times on the Simulator sheet. She can now modify and rearrange any of the tasks on the Task sheet to better balance the workload. It is best to have tasks that can be completed in a week, as no progress is shown until a task is completed. For example, she can split a programming task of six hours, to two tasks of three hours each. Be sure to change the estimated sizes. Also, run the 'Split Plan Time' after completing the changes. Once she is satisfied with the proposed schedule, she is ready to begin the simulation.

Go to the Simulator page and click on the 'Simulate next week activities' button. The week will be simulated and the results posted to the project. The 'Actual phase totals (so far)' on the Simulator page will be updated with the hours expended by phase and LOC generated (if any). The Task sheet will also be updated with the Actual Hours and the Actual Date (if the task was completed.) The Project sheet graphs will also be updated. Any defects removed will be logged to the LOGD sheet. The defects remaining will show in the Project Summary area of the Simulator sheet. It is normal for the defects remaining to approach 100 or more during a simulation. Future tasks should detect and remove them if she has assigned enough review or test time to them. After each weekly simulation, you can adjust task assignments for any task that has not been completed. Be sure to rerun the 'Split Plan Time' if any changes are made before simulating the next week. Continue making adjustments and simulating weeks until you get the 'Simulation Complete!' message. If you run out of time or attempt to simulate a week which you have not assigned any time, you will get an error message. Return to the Schedule sheet, add more time and rerun 'Split Plan Time'. You may also see messages that are displayed in the 'Message

Area’ of the Simulator sheet during the progress of the project. They are informational only and are determined by the number of currently outstanding defects.

RESEARCH VALIDATION

The simulator validation was accomplished by comparing the simulator results to the results of an actual project that was completed by 22 student teams at Embry Riddle Aeronautical University (ERAU) over a three year period. The project consisted of a Flight Planning and Management (FPM) system that maintained minimum separation of aircraft in a variable size "en-route" sector of airspace. The sector is a three dimensional airspace region bounded vertically on the bottom by flight level 180 (FL180 or 18,000 feet), and on the top by flight level 600 (FL600 or 60,000 feet) and horizontally by a rectangle that can vary in size from 50 nautical miles to 150 nautical miles on each side. For more details, see the FPM Needs Statement in the Appendix. Results for the 22 student teams were averaged and standard deviations and other statistics calculated. The metrics used and their units are: Plan Effort (hours), Actual Effort (hours), Plan New and Changed Lines of Code (LOC), Actual New and Changed LOC (LOC), and Defects found and removed (Count). The task schedule was used as input to the simulator, keeping the phase effort distribution and means the same as the ERAU. The simulator was then run 22 times and the results compared to the actual project results. The results of both the ERAU and the simulator are shown in Table 1 and Table 2 respectively.

ERAU TSP Data - Summary - 1999 - 2001

Team	Plan Effort (hrs)	Actual Effort (hrs)	Plan N&C LOC	Actual N&C LOC	Total Defects	Def / KLOC	LOC / Hour
1	209	217	800	612	43	70	2.8
2	178	137	70	269	18	67	2.0
3	125	91	400	1112	59	53	12.2
4	101	69	200	472	55	117	6.8
5	177	163	330	870	50	57	5.3
6	165	128	200	380	35	92	3.0
7	207	228	322	1015	54	53	4.5
1	118	72	450	224	37	165	3.1
2	131	59	580	724	32	44	12.3
3	181	88	700	578	41	71	6.6
4	163	77	550	446	62	139	5.8
5	211	103	1600	850	100	118	8.3
6	179	128	420	356	78	219	2.8
7	190	112	465	865	44	51	7.7
1	315	268	235	418	89	213	1.6
2	288	233	260	540	26	48	2.3
3	123	136	160	240	30	125	1.8
4	204	144	175	661	40	61	4.6
5	177	150	390	911	94	103	6.1
6	134	162	207	325	45	138	2.0
7	213	132	320	438	20	46	3.3
8	189	84	207	325	35	108	3.9
avg	181	136	411	574	49	98	4.9
std dev	51	58	323	266	23	52	3.1
Min	101	59	70	224	18	44	2
Max	315	268	1600	1112	100	219	12

TABLE 1

Simulator TSP Data – Summary – 2005

Team	Plan Effort (hrs)	Actual Effort (hrs)	Plan N&C LOC	Actual N&C LOC	Total Defects	Def / KLOC	LOC/ Hour
1	181	154	411	923	58	63	6.0
2	181	184	411	1034	68	66	5.6
3	181	148	411	298	58	195	2.0
4	181	121	411	252	45	179	2.1
5	181	204	411	949	72	76	4.7
6	181	163	411	408	67	164	2.5
7	181	63	411	248	31	125	3.9
8	181	216	411	672	69	103	3.1
9	181	147	411	344	53	154	2.3
10	181	90	411	224	30	134	2.5
11	181	196	411	1050	60	57	5.4
12	181	250	411	779	69	89	3.1
13	181	126	411	268	46	172	2.1
14	181	201	411	554	63	114	2.8
15	181	114	411	554	39	70	4.9
16	181	175	411	821	50	61	4.7
17	181	102	411	390	29	74	3.8
18	181	133	411	504	35	69	3.8
19	181	201	411	1275	54	42	6.3
20	181	87	411	637	27	42	7.3
21	181	221	411	676	56	83	3.1
22	181	170	411	677	51	75	4.0
avg	181	158	411	615	51	100	3.9
std dev	N/A	49	N/A	302	14	47	1.5
Min	181	63	411	224	27	42	2.0
Max	181	250	411	1275	72	195	7.3

TABLE 2

A summary of the statistical results from the above two tables follows:

Statistical Summary Table

Variable Unit	Statistics	ERAU (N=22)	Simulator (N=22)
Actual Effort Hrs	Mean	136	158
	StdDev	58	49
Actual Effort Hrs	Min	59	63
	Max	268	250
Actual N&C LOC	Mean	574	615
	StdDev	266	302
Actual N&C LOC	Min	224	224
	Max	1112	1275
Defects Count	Mean	49	51
	StdDev	23	14
Defects Count	Min	18	27
	Max	100	72

TABLE 3

There are several statistical tests available for analyzing groups of data. Each test has a set of conditions that must be true for the test to be appropriate. The non-pooled t-test was selected because all of the conditions for its use were met. [Weiss 2005, p.555] Each statistical test has an associated null hypothesis. The null hypothesis for this research states that there is no statistical difference between the means of the ERAU group and the simulator group. There are two approaches available for the t-test; the P-value approach and the critical-value approach. Both are used in this analysis. The P-value is a statistical measure of the probability of getting a value of the test statistic as extreme or more extreme than that observed by chance alone assuming that the null hypothesis is true. A typical significance level used in industry for comparison with the P-value is 0.05. Values less than 0.05 would indicate strong evidence that the null hypothesis should be rejected.

The t-test uses a measure of sample size called 'Degrees of Freedom' (DF) which equals the sum of the sample sizes minus two (22 + 22 – 2) = 42. The DF is used to determine the critical value used in the 'critical-value' approach. The t-test can be 1-tailed or 2-tailed depending on how the hypothesis is stated. If the hypothesis uses 'less than' or 'greater than', then the 1-tailed test would be performed. If the hypothesis uses 'equal' or 'not equal', then the 2-tailed t-test is done. The critical-value is the threshold that the t-statistic is compared with to determine if the null hypothesis should be rejected. The critical-value for a two tailed t-test with 42 degrees of freedom and at the 0.05 significance level is 2.02. If the t-statistic exceeds the 2.02 value the null hypothesis is rejected. The t-statistic comparing the actual effort from the ERAU group and the simulator group is 1.36. The t-statistic indicates that no statistical difference was found between the means. Comparing the actual LOC from ERAU and the simulator, the t-statistic is 0.48. The t-

statistic again shows that the means of the two groups are not statistically different. The P-value for the actual effort is 0.18 and the P-value for the actual LOC is 0.63. These p-values are considered quite large (greater than 0.05) and indicate strong statistical evidence for the null hypothesis that the means are not statistically different. It can be concluded from both approaches that the means of the actual effort and LOC from the ERAU group and the simulator group are not statistically different.

A scatter plot shows the range and distribution for the variables LOC and actual effort. The colors (shading) show the results of actual LOC and effort plotted for both the ERAU and Simulator on the same chart. The scatter plot is another way of showing that the data ranges of the ERAU and simulator data overlap considerably and are similar.

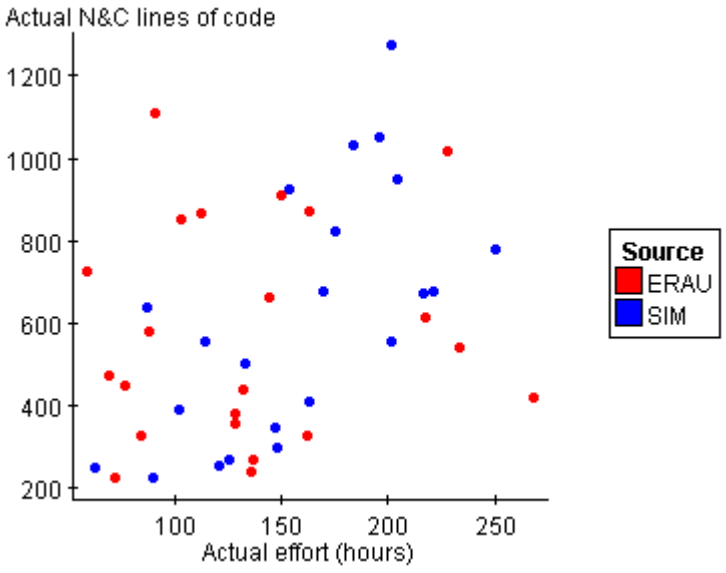

FIGURE 1

CONCLUSIONS

- This research showed that it is possible to create a validated model of the TSPi process that can be used for interactive role playing. Using the metrics available from a simple PSP course, students are able to simulate a project and obtain reasonable and realistic results.
- The planned schedule used by TSPi is sufficient input to a simulator for conducting a realistic simulation.
- The defect insertion and correction rates obtained from Humphrey are accurate enough to model educational projects.
- A PSP course gathers enough data that can be analyzed to produce sufficient information to simulate a TSPi based process.
- This research additionally showed that a realistic model can be prototyped using a spreadsheet.

FUTURE WORK

- This prototype version is calibrated for student use. By using experience factors, similar to the ones used by COCOMO, the metrics could be adjusted to take more experienced team members production into account.
- Evaluate the tool in an actual classroom setting to determine if student process performance is improved.
- Add cost factors so the total cost of a project could be estimated.
- Modify the simulator to generate a sample initial task schedule given only the means of planned effort and planned new and changed LOC. The system could use the industry phase percentages in the calculations.
- The TSPi process has risk assessment features that were not utilized in this simulation. Add risk assessment components to the simulator.
- Modify the simulator so each team member could simulate their own activities each week and then combine them into the team workbook just like the TSPi workbook.
- Verify that the results presented by the simulator are useful to the team leader in learning about process and making changes to improve the schedule.
- Conversion of the simulator to the Access based version when it becomes available.
- Allow other processes to be entered into the simulator replacing TSP.
- Validation of the simulator in an industrial setting.

REFERENCES

[Abdel-Hamid et al 1991] Abdel-Hamid, T., and S. Madnick. 1991. *Software Project Dynamics: An Integrated Approach.* Prentice Hall.

[Baker et al 2003] Alex Baker, Emily Oh Navarro and Andre van der Hoek, 2003. Problems and Programmers: An Educational Software Engineering Card Game.

[Beck 2000] Beck, K. 2000. *eXtreme Programming Explained.* Addison Wesley.

[Boehm 1988] Boehm, B. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer,* May, 61-72.

[Borstler et al 2002] Borstler, J., Carrington, Hislop, Lisack, Olson, Williams, Teaching PSP: Challenges and Lessons Learned. IEEE Software Sep/Oct 2002 pp 42-48.

[Burke 1997] Burke, S. 1997. Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization. Technical Report SEI-96-TR-023. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[Christie 1999] Christie, A. 1999. Simulation -- an enabling technology in software engineering. http://www.sei.cmu.edu/publications/articles /christieapr1999/christie-apr1999 .html

[CMMI 2002] Software Engineering Institute (SEI), Capability Maturity Model Integration, Addison Wesley 2002.

[Coad 1999] Coad, P. 1999. *Java Modeling Color with UML: Enterprise Components and Process.* Prentice Hall.

[Cockburn 2000] Cockburn, A. 2000. Crystal Clear: A Human-Powered Methodology for Small Teams. http://members.aol.com/humansandt/crystal/clear.

[Crosby 1979] Crosby, P. 1979. *Quality Is Free: The Art of Making Quality Certain.* McGraw-Hill.

[Cusamano and Selby 1997] Cusamano, M, and R. Selby. 1990. How Microsoft Builds Software *Communications of the ACM,* June, 53-61.

[Deininger and Schneider 1994] Deininger, M., and K. Schneider. 1994. Teaching Software Project Management by Simulation: Experiences with a Comprehensive Model. *Proceedings of the 7th Conference on Software Engineering Education.* San Antonio TX. pp 227-235.

[Deming 1986] Deming, W. 1986. *Out of the Crisis.* MIT Center for Advanced Engineering Study, Cambridge MA.

[Drappa and Ludewig 2000] Drappa, A; Ludewig, J.: Simulation in Software Engineering Training. *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000).* Limerick, Ireland. pp. 199-208

[Forester 1961] Forester, J. 1961. *Industrial Dynamics.* MIT Press.

[Fowler 2000] Fowler, M. 2000. Put your process on a diet. *Software Development,* 8,12. pp. 32-36.

[Henderson-Sellers 1990] Henderson-Sellers, B. and J. Edwards 1990. The Object-oriented systems life cycle. *Communications of the ACM,* September, 142-59.

[Hilburn 1999] Hilburn, T. 1999. PSP metrics in support of software engineering education. *Proceedings of the 12th Conference on Software Engineering Education and Training.* New Orleans, LA pp. 135-140.

[HPS 1999] HPS.1999. *Process Improvement.* High Performance Systems, Inc.Hanover, NH.

[Humphrey 1995] Humphrey, W. 1995. *A Discipline for Software Engineering.* Addison Wesley.

[Humphrey 1998] Humphrey, W. 1998. Why don't they practice what we preach? *Annals of Software Engineering* 6. pp. 201-222

[Humphrey 2000] Humphrey, W. 2000. *Introduction to the Team Software Process.* Addison Wesley.

[Humphrey 2005] Humphrey, W. 2005, PSP, A Self-Improvement Process for Software Engineers. Addison Wesley.

[IEEE 1997] IEEE. 1991. *IEEE Standard for Developing Software Life Cycle Processes.* Std 1074-1991. Institute of Electrical and Electronics Engineers, New York NY.

[ISO 2001] ISO. 2001. *Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software.*

[Juran 1988] Juran, J. *Juran on Planning for Quality.* Macmillan.

[Laitinen et al 2000] Laitinen, M, M. Fayad, and R. Ward. 2000. Software Engineering in the small. *IEEE Software.* September/October pp. 75.77.

[McFarlan 1974] McFarlan, F. 1974. Effective EDP Project Management. *Managing the Data Resource Function.* R. Nolan, Ed. West Publishing Co.

[McKeen 1981] McKeen, J. 1981. An Empirical Investigation of the Process and Product of Application System Development. PhD Dissertation. University of Minnesota MN.

[Merrill and Collofello 1997] Merril, D., and J. Collofello. 1997. Improving Software Project Management Skills Using a Software Project Simulator. *Proceedings of the Frontiers in Education Conference.* Pittsburgh PA, Session SC3. URL: http://www.engrng.pitt.edu/~fie97

[Navarro 2005] Navarro, E., 2005 *A Survey of Software Engineering Educational Delivery Methods and Associated Learning Theories*, Institute for Software Research, University of California, Irvine.

[Navarro et al 2005] Navarro, E., Andre van der Hoek, Software Process Modeling for an Educational Software Engineering Simulation Game, Software Process Improvement and Practice, upcoming issue (to appear) http://www.ics.uci.edu/~emilyo/

[Paulk et al 1993] Paul, M., C. Weber, S. Garcia, M. Chrisis, M. Bush. 1993. Key Practices of the Capability Model, Version 1.1. Report CMU/SEI-93TR-25. Software Engineering Institute, Carnegie Mellon University, Pittsburgh P A.

[Raffo 2003] Raffo, David, 2002-2003, *Developing Rapidly Deployable Software Process Models*, National Science Foundation.

[Robinson 2005] Robinson, S. Discrete-event Simulation: from the pioneers to the present, what next? Journal of the Operational Research Society, 2005, 56 pp619-629.

[Royce 1970] Royce, W. 1970. Managing the Development of Large Software Systems: Concepts and Techniques. *1970 WESCON Technical papers*. Western Electric Show and Convention, Los Angeles CA. pp A/1-1A1/9.

[Ros et al 2004] *Simulation-based Software Process Modeling and Evaluation,* Islana Rus, et al.

[Schwaber 2000] Schwaber, K. 2000 Scrum. http://www.controlchaos.com/

[Tausworth and Lyu 1996] Tausworth, R and M. Lyu, 1996 A Generalized Technique for Simulating Software reliability. IEEE Software, March, 77-88.

[Thayer 1979] Thayer, R. 1979. Modeling a Software Engineering Project Management System. PhD Dissertation, University of California, Santa Barbara CA.

[Umphress et al 1995] Umphress, D., Helbing, J. Russell, and C. Keen. 1995 Software process maturation. *Information Systems Management 12*, 2. pp. 32-42.

[Umphress et al 2002] Umphress, D., Hendrix, Cross, Software Process in the Classroom: The Capstone Project Experience. IEEE Software Sep/Oct 2002, pp 78-85.

[Weiss 2005] Weiss, Neil A., 2005, Introductory Statistics, Seventh Edition, Addison Wesley.

APPENDICES

APPENDIX A - TSPISIM INSTRUCTION SUMMARY

General Information

Most worksheets are in the 'Protected' mode. To 'Unprotect' a Worksheet, click on the 'Tools' main menu item, then 'Protection', then 'Unprotect sheet…'. No password will be needed. Do not 'Protect' the sheet with a password!

There is a 'ResetSim' button on the 'Task' page to the right of the 'Split Plan Time' button. Click on it and a message box will verify that you want to 'Reset' the simulator. You must enter 'Yes' or no action will be taken. This button clears all the 'Actual' times on the Task sheet and the 'Planned' times on the Schedule sheet. The Planned tasks are not deleted. A message box will verify if the Simulator was reset.

Instructor's Section

Enter list of candidate engineer's on the 'Candidates' page. You will need: Name, initials (must be unique for entire candidate list!), Phone/contact number, eMail address, preferred programming language and the following data from a PSP course: LOC/Hr, Err/KLOC, Pages/Hr, Err/Pg.

Next, update (if desired) the minimum/maximum ranges for the random number calculations on the 'Ranges' page. The min/max are expressed in percentages, so if a task were estimated to take 4 hours, and the min/max percentages were 80/120, then the task could range from 3.2 to 4.8 hours.

Then, update the probability of completion in estimated time and the incremental percent to add if the task is not completed. Example: if the completion probability is 75 and the increment is 10, 75 percent of tasks are completed in the estimated time within the Min/Max range. The rest have 10% added and are tested again. This continues until the task is completed.

Last, update the 'system messages' on the 'Ranges' page to more accurately reflect defects that could occur in the specific project. This step is optional. If you do not want messages to be shown, set the 'Message Frequency' on the Range sheet to a number greater that the project duration.

Student's Section

Fill in the following items on the 'Project' page: Name, Project, Team, Start Date (Note: The Start Date should be a Monday and must be far enough in the past so the project can be completed before the actual current date. The system will not do certain functions using a date in the future!), Instructor, and Cycle.

Select four or five team member's from the 'Candidates' page by entering an 'X' in the Select column. Press the 'Select Team' button and verify that your team members have been correctly transferred to the Team page.

Go to the 'Roles' page and assign team members to their roles by selecting their initials from the drop down box. Verify that each role has been assigned to a valid team member's initials.

Go to the 'SUMS' page and enter the components of the system along with estimated 'Planned Size'. The 'Size Measure' should be 'LOC'. The Base, Deleted, Modified, and Added should be entered. The 'Base' can be zero unless the project is starting with a previous project or 'skeleton' code that is being enhanced. If 'Base' is zero, then all items except 'Added' should be zero. 'Reused' will usually be zero unless there is a Reuse library available for the project.

Go to the 'Task' page and click on the 'Generate Task List' button. Click 'OK' only if you are sure you want to generate a new task list. This will generate a standard default list of tasks for the project.

Assign 'Plan Hours by Role' and Estimated Size and Size Measure. Enter the hours for each team member on the same row under the appropriate role column. Then enter the total size in Pages or LOC for the current task row. Valid Size Measures are: 'LOC' and 'Page'. 'LOC' should be used for Coding tasks only! (Includes Code Reviews and Compile.) All other items should use 'Page'. The task size measure may be left blank if no written output is expected. (Like: Management or PM) You should add, rearrange, or delete task lines as needed during the simulation. Be sure to add/rearrange/delete **full row**s when making changes. DO NOT delete any **columns!**

After verifying that the assignments are evenly distributed by looking at the totals for each engineer shown above each engineer's column, go to the 'Schedule' page and enter the 'Planned Hour' column with weekly team hours. The 'Difference' cell will indicate additional hours needed.

Return to the 'Task' page and Click the 'Update Task and Schedule Plan' button. If you have scheduled enough hours, the last four columns of the 'Task' page will be filled out and the dates will indicate the week each task is scheduled to be completed. An error message will be shown if you have not entered enough hours.

Now click the ‘Split Plan Time’ button. The ‘Task’ page will be split so each row represents a single task for an engineer. The ‘Estimated Size’ will be divided by the number of active engineers assigned to that task row. The ‘Rate’ and ‘Estimated Hours’ columns will also be filled in using the team member’s individual metrics. Verify that the proper amount of time has been allocated for each task by comparing ‘Plan Hours’ against ‘Estimated Hours’. Go to the ‘Simulator’ page and look at the ‘Team Member Phase Totals’. Verify that the work load is evenly distributed. You can modify the assignments to better distribute the work. Note the ‘Team Ratios’ section on the bottom, left of the ‘Simulator’ page. If the ratios are not close enough to the desired Goals, modify tasks as needed and click the ‘Split Plan Time’ button again. You can repeat the process of making adjustments and clicking ‘Split Plan Time’ button as needed. Clicking the ‘Split Plan Time’ button also updates the ‘Team Member Phase Totals’ on the ‘Simulator’ page and recalculates the Schedule.

You can verify that you have a balanced plan by looking at the Quality Profile graph on the Team sheet. Any category that is below 0.8 should be verified. If there is not enough time allocated to the review categories, a substantial number of defects could remain in your project at the completion.

You are now ready to begin the simulation! Click the ‘Simulate the Next Week’s Activities’ button on the ‘Simulator’ page. You should see the first weekly Defect Summary appear. You can also look at the ‘Task’ or the ‘Week’ pages to see what tasks were completed and how much time they actually took. You can make adjustments to any task that has not been completed, then click the ‘Split Plan Time’ button, and verify the modified phase totals on the ‘Simulator’ page. Then click the ‘Simulate the Next Week’s Activities’ button and see the next week’s results. Roughly every two weeks (or interval determined by your instructor), a message could appear on the top right of the ‘Simulator’ page indicating possible problems that needs to be addressed.

Continue simulating each week until you get a message that says the Simulation is Complete! If you get a message stating that no time was assigned for the week, it means that the project did not complete in the weeks allocated. Go to the Schedule sheet and add hours to the next week. Then return to the Simulator page. Good luck.

APPENDIX B - SCREEN SHOTS AND DESCRIPTIONS

The first worksheet that needs to be completed is the Candidates worksheet. It lists the available engineers and their metrics from PSP. The instructor can populate this sheet, or allow the team members to enter their own data. The metrics are copied to the Task sheet to assist with the task planning.

Select Team — Select five members for your team! Then press 'Select Team' button. (Only first five will be used.)

Select	Name	Initials	Phone	eMail	****** Statistics from PSP Exercises ***** Language	LOC/Hr	Err/KLOC	Pgs/Hr	Err/Pg
	Sam Jones	SLJ	706-327-0611	Jones_Samual@colstate.edu	C++	21	21	4.0	1.0
X	William I Westmorland	WIW	706-565-4044	Westmorland_William@colstate.edu	Java	31	18	5.0	2.0
	Susan A Abercrombie	SJA	404-767-9876	Abercrombie_Susan@colstate.edu	C#	25	23	4.0	1.5
X	Julie Craig	JAC	415-327-3431	JCraig@gmail.com	C#	29	26	4.0	1.0
	Sylvia Vorhees	SLV	405-412-2321	SVorhees@gmail.com	Java	28	22	5.0	2.0
	Greg Jameson	GJJ	510-323-2436	GregJameson@gmail.com	C++	16	27	4.0	1.0
X	Gail Muskar	GRM	415-436-3271	GMuskar@gmail.com	C++	22	19	4.0	2.0
X	Robert Green	RAG	334-259-2148	BobGreen@gmail.com	Java	30	20	4.0	1.0
X	Richard Avanzino	RHA	415-756-8639	DickAvanzino@gmail.com	C#	21	31	5.0	1.5
	Carley Simon	CAS	532-294-8639	CarleySimon@gmail.com	COBOL	32	4	4.0	1.0

Candidates Worksheet

After the team assignments are completed following the normal TSP procedures, the next sheet to be filled in is SUMS. The major components of the project are entered along with their planned quantities. For a new project, only the 'Added' column is entered. The SPS and HLD is used for documentation only and have no effect on the project. Only major components should be entered for small projects as several tasks are created for each entry.

ID	Assembly, Sub-Assembly, or Part Name	(A)ssembly or (P)art	Parent Assembly Name	Owner	Size Measure	Base	Deleted	Modified	Added	Reused	New and Changed	Total
1	SRS	A	SYSTEM	JAC	Req Pages	0	0	0	12	0	12	12
2	HLD	A	SYSTEM	JAC	HLD Pages	0	0	0	20	0	20	20
3	Input Module	A	SYSTEM	JAC	LOC	0	0	0	90	0	90	90
4	Process Module	A	SYSTEM	JAC	LOC	0	0	0	261	0	261	261
5	Output Module	A	SYSTEM	JAC	LOC	0	0	0	60	0	60	60

SUMS Worksheet

The Task worksheet is initially generated by clicking the Generate Task List button. Several tasks are generated for each entry in the SUMS sheet except for the SRS and HLD documents. The team leader enters each engineer's assignments for each task by entering the planned time. For tasks that have a deliverable, the unit of measure for the deliverable (Page or LOC) is also entered along with the planned size. If both the

Unit of Measure and Planned Size is entered, the Estimated Hours is calculated to help with the planning. The simulator uses the greater of the Estimated Hours and the Plan Hours.

TSPi Task Planning Template - Form TASK

Name: CPSC4176 | Team: Exotics | Date: 1/6/2005 | Instructor: NLR | Cycle: 1

Split Plan Time | Reset Sim | 181.0 | Total Actual H

Reminder: If Size and Rate are present, estimated hours is calculated as Size / Rat whenever the plan is updated. To prevent calculation, size or rate must

Total Hours by Role: 36.2 36.2 36.2 36.2 36.2 Proj Tot 181

Plan Hours by Role

Generate Task List | Update Task and Schedule Plan

Assembly	Phase	Task	Team Leader	Development Manager	Planning Manager	Quality/Process Manager	Support Manager	Total Resource Hours	Estimated Size	Size Measure	Rate (per Hr.)	Estimated Hours	Plan Hours	Plan Date	Plan Week	Actual Hours	Actual Date
SYSTEM	MGMT	SYSTEM Management and Miscellaneous	0.5					0.5					0.5	1/6/2005	1	0.0	
SYSTEM	MGMT	SYSTEM Management and Miscellaneous		0.5				0.5					0.5	1/6/2005	1	0.0	
SYSTEM	MGMT	SYSTEM Management and Miscellaneous			0.5			0.5					0.5	1/6/2005	1	0.0	
SYSTEM	MGMT	SYSTEM Management and Miscellaneous				0.5		0.5					0.5	1/6/2005	1	0.0	
SYSTEM	MGMT	SYSTEM Management and Miscellaneous					0.5	0.5					0.5	1/6/2005	1	0.0	
SYSTEM	STRAT	SYSTEM Launch and Strategy					2	2					2.0	1/6/2005	1	0.0	
SYSTEM	STRAT	SYSTEM Launch and Strategy				2		2					2.0	1/6/2005	1	0.0	
SYSTEM	STRAT	SYSTEM Launch and Strategy			2			2					2.0	1/6/2005	1	0.0	
SYSTEM	STRAT	SYSTEM Launch and Strategy		2				2					2.0	1/6/2005	1	0.0	
SYSTEM	STRAT	SYSTEM Launch and Strategy	2					2					2.0	1/6/2005	1	0.0	
SYSTEM	PLAN	SYSTEM Planning	2.8					2.8	5	Page	5.0	1.0	2.8	1/6/2005	1	0.0	
SYSTEM	PLAN	SYSTEM Planning		2.8				2.8	5	Page	4.0	1.3	2.8	1/6/2005	1	0.0	
SYSTEM	PLAN	SYSTEM Planning			2.8			2.8	5	Page	4.0	1.3	2.8	1/6/2005	1	0.0	
SYSTEM	PLAN	SYSTEM Planning				2.8		2.8	5	Page	4.0	1.3	2.8	1/6/2005	1	0.0	
SYSTEM	PLAN	SYSTEM Planning					2.8	2.8	5	Page	5.0	1.0	2.8	1/13/2005	2	0.0	
SYSTEM	REQ	SYSTEM Requirements					2	2	5	Page	5.0	1.0	2.0	1/13/2005	2	0.0	
SYSTEM	REQ	SYSTEM Requirements				2		2	5	Page	4.0	1.3	2.0	1/13/2005	2	0.0	
SYSTEM	REQ	SYSTEM Requirements			2			2	5	Page	4.0	1.3	2.0	1/13/2005	2	0.0	
SYSTEM	REQ	SYSTEM Requirements		2				2	5	Page	4.0	1.3	2.0	1/13/2005	2	0.0	
SYSTEM	REQ	SYSTEM Requirements	2					2	5	Page	5.0	1.0	2.0	1/13/2005	2	0.0	
SYSTEM	STP	SYSTEM System Test Plan	1.8					1.8	3	Page	5.0	0.6	1.8	1/13/2005	2	0.0	
SYSTEM	STP	SYSTEM System Test Plan		1.8				1.8	3	Page	4.0	0.8	1.8	1/13/2005	2	0.0	

Task Worksheet (After Split Plan Time' clicked)

Next the Team Leader enters the team schedule on the Schedule worksheet. Total time for the team is entered for each active week of the project. The total hours must be as least equal to the Task Plan hours or the schedule will not be created. After the Schedule entries are completed, the Team Leader returns to the Task worksheet and clicks on the Split Plan Time button. The schedule is created and then separate task rows are created for each engineer for each task. The team leader can now rearrange the tasks in the order that she wants them completed. After the team leader is satisfied with the plan, the simulation is ready to begin. There are summaries of the plan on the Simulator sheet giving helpful information like hours per phase and percent of time in phase. The Team Leader can continue to make changes until she is satisfied with the plan. After any changes, the Split Plan Time is run to update the Plan data on the simulator sheet.

TSPi Schedule Planning Template - Form SCHEDULE

Name	CPSC4176		
Team	Exotics	Total Task Plan Hours	181.0
Date	1/6/2005	Total Schedule Plan Hours	200.0
Instructor	NLR	Difference	-19.0
Cycle	1		

Date	Week	Planned Hours	Cumulative Planned Hours	Actual Hours	Cumulative Actual Hours	Planned Value	Cumulative Planned Value	Earned Value	Cumulative EV
1/6/2005	1	25.0	25.0	25.0	25.0	13.1	13.1	10.0	10.0
1/13/2005	2	25.0	50.0			14.4	27.5		
1/20/2005	3	25.0	75.0			12.0	39.5		
1/27/2005	4	25.0	100.0			15.7	55.2		
2/3/2005	5	25.0	125.0			13.6	68.9		
2/10/2005	6	25.0	150.0			12.9	81.8		
2/17/2005	7	50.0	349.0			18.2	100.0		

Schedule Worksheet (After schedule generated)

This is the Simulator after the first week has been run. Note the plan data on the left and the 'Actual' results in the center. The right side is weekly defect summary. There are currently 18 undiscovered defects.

Simulator Metrics and Message Area

Simulate next week activities

Next Week # 2

Message Area:

	Team member planned phase totals							Actual phase totals (so far)						
Phase	Total/%	TL	DM	PM	QPM	SM	Total/%	TL	DM	PM	QPM	SM
# Tasks		25	25	25	25	25		2	3	2	2	3
Tot Hours	181.0	36.2	36.2	36.2	36.2	36.2	25	5	5	5	5	5
Mgmt/Anal	8.8%	3.2	3.2	3.2	3.2	3.2	9.2%	0.4	0.5	0.4	0.6	0.4
Strat/Plan	13.3%	4.8	4.8	4.8	4.8	4.8	83.6%	4.6	3.9	4.6	4.4	3.4
REQ	10.5%	3.8	3.8	3.8	3.8	3.8	7.2%	0	0.6	0	0	1.2
REQINSP	3.9%	1.4	1.4	1.4	1.4	1.4	0.0%	0	0	0	0	0
HLD / ITP	8.3%	3	3	3	3	3	0.0%	0	0	0	0	0
HLDINSP	2.2%	0.8	0.8	0.8	0.8	0.8	0.0%	0	0	0	0	0
DLD / TD	11.3%	4.1	4.1	4.1	4.1	4.1	0.0%	0	0	0	0	0
DLDR	5.0%	1.8	1.8	1.8	1.8	1.8	0.0%	0	0	0	0	0
DLDINSP	1.4%	0.5	0.5	0.5	0.5	0.5	0.0%	0	0	0	0	0
CODE	9.4%	3.4	3.4	3.4	3.4	3.4	0.0%	0	0	0	0	0
CR	4.7%	1.7	1.7	1.7	1.7	1.7	0.0%	0	0	0	0	0
COMPILE	1.4%	0.5	0.5	0.5	0.5	0.5	0.0%	0	0	0	0	0
CODEINSP	1.4%	0.5	0.5	0.5	0.5	0.5	0.0%	0	0	0	0	0
UT	1.4%	0.5	0.5	0.5	0.5	0.5	0.0%	0	0	0	0	0
IT	6.1%	2.2	2.2	2.2	2.2	2.2	0.0%	0	0	0	0	0
ST	8.3%	3	3	3	3	3	0.0%	0	0	0	0	0
MISC / DOC	2.8%	1	1	1	1	1	0.0%	0	0	0	0	0
LOC	411	90	100	60	101	60	0	0	0	0	0	0

Project Summary			
Totals	18	0	18
Week Number	Defects Added	Defects Removed	Defects Remaining
1	18	0	18

Weekly Defect Summary

Simulator Sheet (After Week 1)

This is the simulator after week 3 has been run. Note that the actual time has progressed and 18 defects (out of 58 inserted) were found and removed.

Actual phase totals (so far)							Project Summary			
Total/%	TL	DM	PM	QPM	SM	k	Totals	58	18	40
	6	7	7	7	9	1				
74.9	15	15	14.9	15	15	2	Week	Defects	Defects	Defects
3.1%	0.4	0.5	0.4	0.6	0.4	3	Number	Added	Removed	Remaining
28.2%	4.7	3.9	4.7	4.4	3.4	4				
31.9%	6.5	5	4.3	4.6	3.5	5	Weekly Defect Summary			
12.8%	2.4	1.6	2	1.9	1.7	6	1	18	0	18
18.3%	1	4	3.5	2.7	2.5	7	2	27	4	41
1.9%	0	0	0	0.8	0.6	8	3	13	14	40
3.9%	0	0	0	0	2.9	9				

Simulator (After Week 3)

The 18 defects are logged to the LOGD worksheet along with the phase each was injected and removed.

TSPi Defect Recording Log - Form LOGD

Name CPSC4176 **Date** 1/20/2005
Team Exotics **Instructor** NLR
7/5/2005 **Cycle** 1

Date	Num	Type	Assembly	Injected	Removed
1/13/2005	1	10	SYSTEM	REQ	REQINSP
1/13/2005	2	10	SYSTEM	REQ	REQINSP
1/13/2005	3	10	SYSTEM	REQ	REQINSP
1/13/2005	4	10	SYSTEM	REQ	REQINSP
1/20/2005	5	10	SYSTEM	REQ	REQINSP
1/20/2005	6	10	SYSTEM	REQ	REQINSP
1/20/2005	7	10	SYSTEM	REQ	REQINSP
1/20/2005	8	10	SYSTEM	REQ	REQINSP
1/20/2005	9	10	SYSTEM	REQ	REQINSP
1/20/2005	10	10	SYSTEM	REQ	REQINSP
1/20/2005	11	10	SYSTEM	REQ	REQINSP
1/20/2005	12	10	SYSTEM	REQ	REQINSP
1/20/2005	13	10	SYSTEM	REQ	REQINSP
1/20/2005	14	10	SYSTEM	REQ	REQINSP
1/20/2005	15	10	SYSTEM	REQ	REQINSP
1/20/2005	16	10	SYSTEM	REQ	REQINSP
1/20/2005	17	10	SYSTEM	HLD	HLDINSP
1/20/2005	18	10	SYSTEM	HLD	HLDINSP

LOGD Worksheet

From the following worksheet, we can see that some tasks are not being completed in the desired sequence because some team members are a bit behind. The Team Leader could reassign some tasks to help the team members catch up.

TSPi Task Planning Template - Form TASK

Name: CPSC4176 | Team: Exotics | Date: 2/10/2005 | Instructor: NLR | Cycle: 1

Split Plan Time | Reset Sim | 181.0 | Total Actual Hours: 149.9

Total Hours by Role: 36.2 36.2 36.2 36.2 36.2

Reminder: If Size and Rate are present, estimated hours is calculated as Size / Rate whenever the plan is updated. To prevent calculation, size or rate must be bl

Plan Hours by Role

Generate Task List | Update Task and Schedule Plan

Assembly	Phase	Task	Team Leader	Development Manager	Planning Manager	Quality/Process Manager	Support Manager	Total Resource Hours	Estimated Size	Size Measure	Rate (per Hr.)	Estimated Hours	Plan Hours	Plan Date	Plan Week	Actual Hours	Actual Date	Actual Week
Output Module	CR	Output Module Code Review	1.7					1.7	60	LOC	171.0	0.4	1.7	2/10/2005	6	1.2		
SYSTEM	MGMT	SYSTEM Management and Miscellaneous	0.5					0.5					0.5	2/10/2005	6	0.0		
SYSTEM	MGMT	SYSTEM Management and Miscellaneous		0.5				0.5					0.5	2/10/2005	6	0.8	2/10/2005	6
SYSTEM	MGMT	SYSTEM Management and Miscellaneous			0.5			0.5					0.5	2/10/2005	6	0.0		
SYSTEM	MGMT	SYSTEM Management and Miscellaneous				0.5		0.5					0.5	2/10/2005	6	0.4	2/10/2005	6
SYSTEM	MGMT	SYSTEM Management and Miscellaneous					0.5	0.5					0.5	2/10/2005	6	0.6	2/10/2005	6
Input Module	COMPILE	Input Module Compile	0.5					0.5					0.5	2/10/2005	6	0.0		
Process Module	COMPILE	Process Module Compile		0.5				0.5					0.5	2/10/2005	6	0.5	2/10/2005	6
Process Module	COMPILE	Process Module Compile			0.5			0.5					0.5	2/10/2005	6	0.0		
Process Module	COMPILE	Process Module Compile				0.5		0.5					0.5	2/10/2005	6	0.5	2/10/2005	6
Output Module	COMPILE	Output Module Compile					0.5	0.5					0.5	2/10/2005	6	0.5	2/10/2005	6
Input Module	CODEINSP	Input Module Code Inspection					0.5	0.5	50	LOC	121.0	0.4	0.5	2/10/2005	6	0.8	2/10/2005	6
Process Module	CODEINSP	Process Module Code Inspection				0.5		0.5	50	LOC	168.0	0.3	0.5	2/10/2005	6	0.4	2/10/2005	6
Process Module	CODEINSP	Process Module Code Inspection			0.5			0.5	50	LOC	180.0	0.3	0.5	2/10/2005	6	0.0		
Process Module	CODEINSP	Process Module Code Inspection		0.5				0.5	50	LOC	155.0	0.3	0.5	2/10/2005	6	0.2		
Output Module	CODEINSP	Output Module Code Inspection	0.5					0.5	50	LOC	193.0	0.3	0.5	2/10/2005	6	0.0		
Input Module	UT	Input Module Unit Test	0.5					0.5	1	Page	5.0	0.2	0.5	2/10/2005	6	0.0		
Process Module	UT	Process Module Unit Test		0.5				0.5	1	Page	4.0	0.3	0.5	2/10/2005	6	0.0		
Process Module	UT	Process Module Unit Test			0.5			0.5	1	Page	4.0	0.3	0.5	2/10/2005	6	0.0		
Process Module	UT	Process Module Unit Test				0.5		0.5	1	Page	4.0	0.3	0.5	2/10/2005	6	0.5	2/10/2005	6
Output Module	UT	Output Module Unit Test					0.5	0.5	1	Page	5.0	0.2	0.5	2/10/2005	6	0.5	2/10/2005	6
SYSTEM	IT	SYSTEM Build and Integration Test					2.2	2.2	1	Page	5.0	0.2	2.2	2/10/2005	6	1.7		

Task Worksheet (After Week 6)

As the project progresses, the schedule worksheet shows the earned value, and the actual time expended for each week.

TSPi Schedule Planning Template - Form SCHEDULE

Name: CPSC4176 | Team: Exotics | Date: 2/17/2005 | Instructor: NLR | Cycle: 1

Total Task Plan Hours: 181.0 | Total Schedule Plan Hours: 200.0 | Difference: -19.0

Date	Week	Planned Hours	Cumulative Planned Hours	Actual Hours	Cumulative Actual Hours	Planned Value	Cumulative Planned Value	Earned Value	Cumulative EV	Predicted Hours	Cumulative Predicted Hours	Predicted Earned Value	Cumulative Predicted Earned Value
1/6/2005	1	25.0	25.0	25.0	25.0	13.1	13.1	10.0	10.0			10.0	10.0
1/13/2005	2	25.0	50.0	25.0	50.0	14.4	27.5	14.9	24.9			14.9	24.9
1/20/2005	3	25.0	75.0	24.9	74.9	12.0	39.5	10.6	35.5			10.6	35.5
1/27/2005	4	25.0	100.0	25.0	99.9	15.7	55.2	11.5	47.0			11.5	47.0
2/3/2005	5	25.0	125.0	25.0	124.9	13.6	68.9	13.4	60.4			13.4	60.4
2/10/2005	6	25.0	150.0	25.0	149.9	12.9	81.8	13.1	73.5			13.1	73.5
2/17/2005	7	50.0	349.0	46.1	196.0	18.2	100.0	26.5	100.0			26.5	100.0

Schedule Worksheet (After project completed)

The Simulator shows summaries of time per phase for each engineer, weekly defect summary, team ratios, defects by phase and consolidated phase summaries.

Team member planned phase totals							Actual phase totals (so far)							Project Summary			
Phase	Total/%	TL	DM	PM	QPM	SM	Total/%	TL	DM	PM	QPM	SM	k	Totals	91	85	
# Tasks		25	25	25	25	25		25	25	25	23	25	1				
Tot Hours	181.0	36.2	36.2	36.2	36.2	36.2	196	42.1	38	41.8	38.8	35.3	2	Week	Defects	Defects	Defe
Mgmt/Anal	8.8%	3.2	3.2	3.2	3.2	3.2	9.5%	3.8	3.4	3.6	4.5	3.3	3	Number	Added	Removed	Remair
Strat/Plan	13.3%	4.8	4.8	4.8	4.8	4.8	10.8%	4.7	3.9	4.7	4.4	3.4	4				
REQ	10.5%	3.8	3.8	3.8	3.8	3.8	12.2%	6.5	5	4.3	4.6	3.5	5	Weekly Defect Summary			
REQINSP	3.9%	1.4	1.4	1.4	1.4	1.4	4.9%	2.4	1.6	2	1.9	1.7	6	1	18	0	
HLD / ITP	8.3%	3	3	3	3	3	8.5%	2.9	4.1	4.4	2.7	2.5	7	2	27	4	
HLDINSP	2.2%	0.8	0.8	0.8	0.8	0.8	2.1%	1	0.6	1.1	0.8	0.6	8	3	13	14	
DLD / TD	11.3%	4.1	4.1	4.1	4.1	4.1	10.3%	3.2	2.8	3.8	4.7	5.7	9	4	15	10	
DLDR	5.0%	1.8	1.8	1.8	1.8	1.8	5.7%	2.3	3.4	2.3	1.4	1.8	10	5	14	24	
DLDINSP	1.4%	0.5	0.5	0.5	0.5	0.5	1.6%	0.9	0.7	0.6	0.6	0.3	11	6	4	11	
CODE	9.4%	3.4	3.4	3.4	3.4	3.4	8.9%	3.8	3.7	4.5	2.3	3.1	12	7	0	22	
CR	4.7%	1.7	1.7	1.7	1.7	1.7	6.4%	3.1	1.7	2.9	2.3	2.5	13				
COMPILE	1.4%	0.5	0.5	0.5	0.5	0.5	1.2%	0.6	0.5	0.3	0.5	0.5	14				
CODEINSP	1.4%	0.5	0.5	0.5	0.5	0.5	1.4%	0.4	0.7	0.4	0.4	0.8	15				
UT	1.4%	0.5	0.5	0.5	0.5	0.5	1.2%	0.5	0.3	0.6	0.5	0.5	16				
IT	6.1%	2.2	2.2	2.2	2.2	2.2	5.2%	1.8	1.5	2.9	1.6	2.4	17				
ST	8.3%	3	3	3	3	3	7.9%	3.5	3.4	1.9	4.7	2	18				
MISC / DOC	2.8%	1	1	1	1	1	2.3%	0.7	0.7	1.5	0.9	0.7	19				
LOC	411	90	100	60	101	60	1468	347	372	218	355	176	20				

Simulator (After project completed)

Interesting Team Ratios

	Goal	Plan	Actual
Design as % of Code	100%	209%	211%
Code Review % of Coc	50%	65%	87%
Des Review % of Desi	50%	44%	50%

Defects by Phase

	Inject	Rem	Left
Plan/Req	47	41	6
Design-High	13	13	0
Design-Detail	21	21	0
Code	10	10	0
Test	0	0	0
Total	91	85	6

Plan and Actual Time in Phase

Phase	**** Plan ****		*** Actual ***	
	Hrs	%	Hrs	%
Mgmt&Analy	16.0	8.8%	18.6	9.5%
Plan	24.0	13.3%	21.1	10.8%
Requirements	26.0	14.4%	33.5	17.1%
Design	24.0	13.3%	25.2	12.9%
Implement	65.0	35.9%	71.9	36.7%
Test	26.0	14.4%	25.7	13.1%
Totals	181.0	100.0%	196.0	100.0%

Phase Definitions:

M&M + PM
STRAT + PLAN
REQ + STP + REQINSP
HLD + ITP + HLDINSP
DLD(all) + TD + CODE(all) _
+ COMPILE + UT
(Test) = IT + ST

Simulator Ratios, Defects by Phase, and Consolidated Phase Summary

The Ranges worksheet is used by the instructor to establish the ranges for the random number function. In the below examples for Documentation, Coding, and Hourly Tasks, the simulator will stay between 50% and 150% of the expected values. For the Defect Factors, the first one is for defects added and the second one is for defects removed. Both are currently set to 50% of their expected values. The Probability of

Completion is used to determine if a task was completed in the calculated time. Currently, 60% of the time a task will be completed. If a task is not completed, the incremental amount is added (10%) and the task is tested again. This process continues until the task is marked as complete. The overtime field will allow team members to work extra time each week if needed to complete their assigned tasks.

Activity	Units	Min %	Max %
Documentation	Page	50%	150%
Coding	LOC	50%	150%
Hourly Tasks	Hours	50%	150%
Defect factor(Add/Rem)	Defects	50%	50%
Probability of Completion in estimated time			60%
Incremental % to add			10%
Max % Overtime Allowed			0%

Ranges Worksheet (Activity Parameters)

The values in the Errors Fixed per Hour section determine minimums and maximums team parameters that are not available for individual team members. The High Level Design values are for minimum, maximum and average production values. A team can productively review from 3.1 to 5 pages of the high level design document per hour. They can effectively review from 100 to 200 LOC per hour, and can review from 0.8 to 2.0 pages of requirements or low level design documents per hour. These values came from Watts Humphrey's books.

The first two project level parameters, Effort and LOC, contain the minimum, maximum, average, and standard deviation of the expected project values. The next three, Design as percent of code, Code review as percent of code, Design review as percent of design are not currently used, but could be used in future work to determine allowable ranges for the plan and issue extra defects if the plan varies too far from the allowed ranges. Similar parameters on the Quality Profile Parameter sheet are used by the Quality Profile graph on the Project worksheet.

The message frequency is used to determine how many weeks pass before issuing a message. A value of seven indicates that a message will be issued every 7 weeks. A value of 1 would result in a message every week. The message depends on the number of defects left in the project. There are three levels of messages that the instructor can customize for a project, Minor, Intermediate, and Severe, each with 10 messages. Each level has a maximum number of defects that determine which level message is issued. Currently, if there are six or fewer defects, a minor message is issued. Up to 15 defects cause an intermediate message and more than 15 cause a severe message

The Cost Factor is used to increase the cost (in time) to correct a defect. For each phase that has passed before a defect is found, the cost is multiplied by the factor. The phases used for this calculation are the ones shown in the Plan and Actual Time in Phase Summary shown on the Simulator worksheet.

Errors fixed per hour	Units	Min	Max	Avg	
High Lvl Design	Page	3.1	5.0	4.1	
Source Code Rev	LOC	100.0	200.0	150.0	
Req & LL Design	Page	0.8	2.0	1.6	
Project Level Parameters		Min	Max	Avg	SD
Effort (Hours)		60	270	135	58
LOC		225	1115	575	266
Design as % of Code		90%	110%	100.00%	
Code Review as % of Code		40%	60%	50.00%	
Design Review as % of Design		40%	60%	50.00%	
Message Frequency (Weeks)			7		
Cost Factor			1.5		

Ranges Worksheet (Project Parameters)

APPENDIX C - FPM NEED STATEMENT

The FPM system provides for automating flight planning and management for aircraft that fly within a single “en-route’ control sector.

Functional Needs

1. AN FPM “en-route” sector is a three-dimensional airspace region with the following characteristics:

    a. The sector is bounded vertically by flight levels FL180 (18,000 feet) and FL600 (60,000 feet.)

    b. The sector is bound horizontally by a rectangle where the vertices and the points inside and outside the polygon are designated with an (x, y) coordinate system. The coordinates x and y are positive real numbers in units of nautical miles. Each side of the rectangle must be at least 50 nautical miles in length and not more that 150 nautical miles in length.

2. The FPM system will allow a maximum of 100 aircraft within a sector at any given time.

3. An aircraft must maintain the same altitude (between FL180 and FL600) from sector entry to sector exit.

4. Five aircraft, at most, can be flying at the same altitude at any given time.

5. All aircraft must, at all times, maintain a minimum separation of at least 0.5 nautical miles. (~3,000 feet ‘straight line’ distance between each pair.)

6. Each aircraft in the system must have an approved flight plan for flying through the sector. The flight plan consists of the following:

    a. an aircraft ID number

    b. the pilot’s name

    c. the amount of fuel on board (in minutes flying time)

    d. an air route that is a straight line path consisting of:

        i) a start point (x, y) and a start time

        ii) a cruising altitude

iii) a cruising speed

iv) a true course

v) an end point and an end time

7. The start point and end point of the path must be points on a sector boundary.

8. Prior to entering the sector an aircraft must submit a flight plan to the sector controller and have it approved. A flight plan for an aircraft will not be approved unless constraints on sector traffic and aircraft separation are satisfied, and all the flight plan data is consistent and correct.

9. The FPM must have a text-based user interface that allows a user to carry out the following operations:

   a. INITIALIZE_SECTOR
   Input the sector characteristics and initialize the sector.

   b. SUBMIT_FP
   Input a flight plan; check the plan for correctness and consistency with currently approved plans. If there are no problems, output an acceptance confirmation message; otherwise, output a message that describes the reason(s) for non-acceptance.

   c. SUBMIT_SET_FP
   Submit a set of flight plans via a text file. The order of the flight plans in the text file will be treated as a sequential, chronological submission of flight plans.
   The text file will have the following format:

Line Number	Contents
1	Aircraft ID number, Pilot's name
2	Amount of fuel on board (minutes)
3	Start point, start time, cruising altitude, cruising speed, true course, end point, end time

An example file:
for sector with vertices (10,20), (100,20), (10,100), (100,100):

Line Number	Contents
1	DAL111, J Doolittle
2	60
3	10.00, 40.00, 1410, 220, 300.00, 39.80, 60.00, 100.00, 1425
4	AF237, N Rogers
5	90
6	80.00, 20.00, 1410, 220, 400.00, 36.30, 20.00, 100.00, 1426

d. DELETE_FP
Input an aircraft ID and delete the flight plan from the FPM system.

e. GET_FP
Input an aircraft ID and output the flight plan for the aircraft.

f. GET_SEC_FP
Input a time, and output the ID's for those aircraft whose flight plans indicate they will be inside the sector at the given time.

g. GET_ALT_FP
Input a time and altitude, and output the ID's for the aircraft whose flight plans indicate they will be in the sector at the given altitude and given time.

h. CHANGE_ALTITUDE
Input an aircraft ID and change the altitude on the flight plan for the aircraft. (Must be changed BEFORE activation of plan!)

i. SEPARATION
Input two aircraft ID's. Output the minimum straight line distance (SLD) between the aircraft and the time(s) when it will occur.

j. SMALL_SEPARATION
Input a distance and output the ID's for those pairs of aircraft whose flight plans indicate the straight line distance (SLD) between them will be less than or equal to the distance entered.

Note: 'Straight Line Distance' (SLD) is defined as the square root of the sum of the squares of the differences of the x, y, and z coordinates **in feet**.

$$SLD = SQRT((X2-X1)^2 + (Y2-Y1)^2 + (Z2-Z1)^2$$

Additional Characteristics / Constraints

10. The following are additional characteristics of the FPM system:

    a. The system will be easy to use. A user will not be required to possess special computer knowledge or in-depth familiarity with aviation terminology.

    b. Clock time will be in 24 hour format. Elapsed time will be measured in minutes. Computed times must be accurate to within one minute.

    c. Direction will be measured in degrees, relative to "true north" (0 to 359.9 degrees). Computations must be accurate to 0.1 degrees)

    d. Horizontal distances, and the x and y coordinates of a position will be measures in nautical miles. Computations must be accurate to within 0.1 nautical miles.

    e. Altitude will be measured in feet, and the flight plan altitudes will be represented in units of hundreds of feet (e.g. FL220 is 22,000 ft.)

    f. All FPM operations should incorporate appropriate exception handling so that the system responds with a clear, descriptive message when an error or exception condition occurs.

Notes:

A nautical mile is 1852 meters or about 6076 feet. For our purposes, we will use 3000 feet as approximately equal to 0.5 nautical miles for separation calculations.

A 'US' mile is 5280 feet or 8 furlongs
A furlong is 40 rods, a rod is 16 ½ feet, a foot is 12 inches.

Non-Functional Needs

11. The following are some non-functional needs of the FPM system:

    a. The system must be developed using the TSPi process.

    b. The system must have a user/reference manual.

    c. The implementation programming language must be a version of a language available in the CSU labs. (JAVA, C++, or C#)

    d. The system must be easily portable to a variety of platforms.

    e. The system must be easy to maintain. (Object or highly modular!)

Special notes:

For the purpose of this project, you may assume that the FPM system is not a "real-time" application. That is, you may assume that all flight plans are submitted (along with any changes) at times before they go into effect.

The customer for this project is Neal Rogers. He may be consulted concerning clarifications, alterations, or additions to the FPM requirements.