A FLEXIBLE MODEL FOR MULTI-AGENT BASED SIMULATION OF SOFTWARE DEVELOPMENT PROCESS

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.

_____

Ravikant Agarwal

Certificate of Approval:

_____	_____
Kai H. Chang Professor Computer Science and Software Engineering	David A. Umphress, Chair Associate Professor Computer Science and Software Engineering
_____	_____
Dean Hendrix Associate Professor Computer Science and Software Engineering	George T. Flowers Interim Dean Graduate School

A FLEXIBLE MODEL FOR MULTI-AGENT BASED SIMULATION OF SOFTWARE DEVELOPMENT PROCESS

Ravikant Agarwal

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
December 17, 2007

A FLEXIBLE MODEL FOR MULTI-AGENT BASED SIMULATION OF SOFTWARE DEVELOPMENT PROCESS

Ravikant Agarwal

_____

Signature of Author

_____

Date of Graduation

DISSERTATION ABSTRACT

A FLEXIBLE MODEL FOR MULTI-AGENT BASED SIMULATION OF SOFTWARE DEVELOPMENT PROCESS

Ravikant Agarwal

Doctor of Philosophy, December 17, 2007
(M.S. Wilkes University, 2004)
(B.S. Wilkes University, 2002)

340 Typed Pages

Directed by David Umphress

Inadequate use of project management techniques in software development can be traced to the lack of efficient education strategies for managers [24]. Software development processes are complex and therefore it is hard to predict how changes made to some part of the process can affect the overall outcome of the process. Introducing change in the process is often time consuming and there is no assurance that the change implemented will result in an improvement. Simulation of software development process provides an easy way for managers to test the different configurations of the process and understand the effects of various policies.

In this work, the users of the simulation can act as managers without caring about the risks of the failures of real software projects. This simulation is a learning-by-doing model where software developers and students can understand the concepts of software process and the underlying risk, and other decision making activities. This project aims at developing a software process simulation with an agent architecture, where the agents are the entities that mimic the real software developers within the simulation. The individual behavior of the agents is based on the statistical data collected over a period of time using Personal Software Process (PSP). Given the statistics collected, expected behavior of the individuals can be enacted over time during the scope of software development.

The provision of having a user constantly monitoring the software development within the simulation is made. In this work, simulation processes can be changed during the model execution. The simulation accepts the feedback from the user after every period of time. The simulation interface displays the actual vs. planned matrix. Based on these matrices of comparison, the user can introduce any changes to the software process that he thinks might improve the overall development cycle of the project.

Using agent directed simulation to mimic the software development process at the individual level also would enable us to introduce a new phase of software development without having to change the simulation code. This simulation would start with a given number of agents initialized by the user. At any point of time, the user may change the number of developers or assign developers on different phases of the software development depending on their performance and capabilities.

Style manual: Software Engineering

Software used: Microsoft Word 2003 SP2

ACKNOWLEDGEMENTS

I gratefully acknowledge my indebtness to my major professor, Dr. David A. Umphress. His guidance, feedback, and sense of humor made this undertaking a wonderful experience. It is only through his constant help and encouragement that the research progressed towards its goal. His vast experience and knowledge provided me with the much needed direction throughout the research work.

I would also like to thank my committee members, Dr. Kai Chang, and Dr. Dean Hendrix for their participation on my committee, as well as their tutelage throughout my graduate study. I would also take this opportunity to thank Dr. Maghsoodloo and Dr. Chapman for helping me through the validation of this research.

Above all, it is a pleasure to acknowledge my family who have given abundantly of their support, guidance, and love throughout my entire life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

This chapter discusses the statement of the problem. It introduces the concept of simulation in software development process to the readers and discusses the need for the same. It also describes the problems present in the current systems that simulate the software development processes.

1.1. Statement of the problem

A software development process is defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products e.g., project plans, design documents, code, test cases, and user manuals. (Figure 1.1) [77].

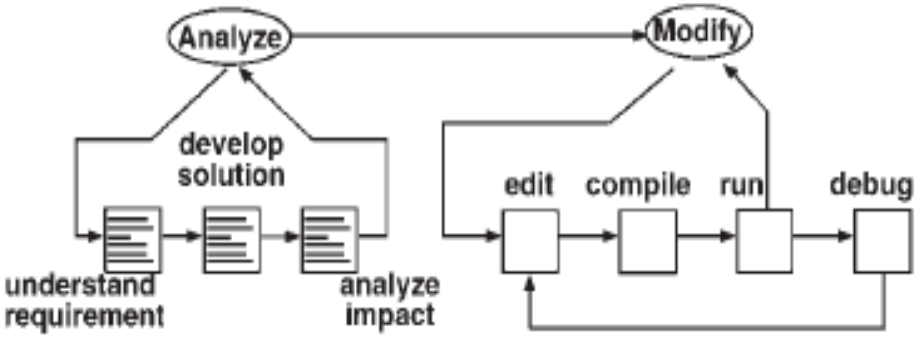

Figure 1.1: A simple software development process depicted as a directed graph [91]

The enactment of a software development process results in a set of artifacts, some of which are produced to support subsequent development activities. The activities and the order in which they will be performed are planned in the early stages of a software development project, usually according to some predefined process. Activities can be, for example, requirements elicitation, architectural analysis, unit test specification, or programming. Most activities take some artifacts as input and produce one or more new or altered artifacts as output.

Wickenberg et al. [102] claim that the software development industry has been failing to meet expectations in terms of cost, quality and schedule. A portion of the community has turned its focus to the software development process in order to reduce cost and schedule time, increase quality and to allow predictions of these variables during a development effort. There is still a significant part of the software community that doesn't embrace the process philosophy.

The software development community is finding ways to study and improve the existing processes. One approach that has been seen to be widely accepted is the simulation of the software processes at different levels. A simulation is an execution of a model, represented in a virtual environment that gives the information about the system being investigated. Simulation and visual rendering of models enable analysts and decision-makers to enhance their decision efficiency, by rehearsing strategies to avoid hidden pitfalls. Software process simulation is becoming increasingly popular in the software engineering community, both among academics and practitioners [56]. An expression of this popularity is the annual ProSim workshop (International workshop on

Software **Pro**cess **Sim**ulation and Modeling) [44], which had its first meeting in 1998. Software Process Simulation has received substantial attention over the last twenty years. But only in recent years it is beginning to be used to address several issues concerning the strategic management of software development and process improvement support. The development of software is a complex activity that involves a large number of factors that define its success. Real-world experimentation is difficult and costly which lead the researchers to use various techniques in an attempt to simulate the development processes.

The focus of a software process simulation model is usually on specific portions of the software development process. A model is a simplification and an approximation of the real system. The model developer designs a model with only certain aspects of the software process which are relevant to address the issues that he is studying.

New and innovative software engineering techniques are being developed constantly, so a better understanding of them is useful to evaluate their effectiveness and to predict possible problems. Simulation can provide information about these issues avoiding real world experimentation, which is known to be too costly in terms of time and money. Simulation can also help project managers and process engineers to plan changes in the development process. The development of a simulation model is an inexpensive way to collect information when costs, risks and complexity of the real system are very high. In order to create a connection between real world and simulation results, it is usual to combine empirical findings and knowledge from real processes. In

general, empirical data are used to calibrate the model, and the results of the simulation process are used for planning, design and analyzing real experiments.

1.1.1. The need for simulating software development process

Software development processes and organizations are complex and it is therefore often hard to predict how changes made to some part of the process or organization will affect other parts and the outcome of the process as a whole. Making changes to a process is often time consuming and generally, only one process change can be performed at a time. In addition, like many processes, software processes can contain multiple feedback loops such as those associated with correction of defects in design or code. Delays that result from these effects may range from minutes to years. The complexity that results from these effects and their interactions makes it almost impossible for human (mental) analysis to predict the consequences. Unfortunately, traditional process analysis does not shed much light on the behavioral issues, and the usual way to resolve them is to run the process and observe the consequences. This can be an extremely costly way to perform process improvement. Moreover, as there is no guarantee that the change imposed on the process is actually an improvement, managers are typically hesitant to experiment.

Simulation provides insights into the complex process behavior. It is a cost-efficient way to allow developers and managers to elaborate different configurations of the process, and understand the effects of various policies. Software development process simulation serves a variety of objectives. It can be used as a method for gaining better understanding of the software development process in general; in these cases, the goal is

to achieve some generally applicable knowledge about some part of the development process. Alternatively, it can be used to better understand a specific project or organization to forecast the outcome as a support for risk management, planning, strategic management or some other decision-making activities.

The reasons for simulating software development processes range from supporting strategic and operational management of software development projects to process improvement and understanding. Let's look at some broad areas where simulation of software development process proves to be useful.

1.1.1.1 Assessing the Costs of Software Development

At an applied level, simulation can support project costing, planning, tracking, and prediction. In a competitive world, accurate prediction provides a significant advantage. If cost estimates are too high, bids are lost; if too low, organizations find themselves in debt. In this context, simulation can provide not only estimates of cost but also estimates of cost uncertainty. Simulation is a powerful tool to aid activity-based costing and can incrementally accumulate costs to an extremely fine degree of resolution. In addition, it can assess the uncertainty of costs based on the interacting uncertainties of independent variables [32, 97].

1.1.1.2 Supporting Metric Collection

Simulation is effective only if both the model and the data used to drive the model accurately reflect the real world. There is a tight connection between the model and the

data in the sense that a simulation can only be executed if it is supplied with numerical drivers, which forces the developer to identify points in the model where these drivers are needed. For example, one set of data that needs to be entered in the model may be what percentage of design documents pass review and what percentage must be returned for further work. Thus, in the construction of the model, points where metric data must be collected fall out as a bonus. This approach forces the collection of metric data in a consistent sense from a systems perspective—it is not merely “nice to have” data. Often, too much or too little metric data is collected because the analyst does not have clear guidelines on what is essential [15].

1.1.1.3 Building Consensus and Communication

When changes to a process are proposed, development experience of the developers is likely to be the most important influence. However, experience may not be enough to correctly assess behavioral changes resulting from process modifications. One person’s experience may not correspond to another’s, and subjective judgment comes into play as to whose opinion is correct. Usually, the person with greater authority wins. With the ability to quantify the effects through simulation, a much greater degree of insight and understanding can be brought to bear on the decision making process. Therefore, simulation can be a significant influence in communication and consensus building. In this context, alternate process designs can be considered in a quantitative manner with respect to such issues as bottlenecking, resource availability, throughput,

and costs. This analysis should result in processes that, once installed, will have a considerably higher probability of satisfactory operation.

1.1.2. What to simulate

In general, the basic purpose of the simulation model, coupled with the specific questions or issues to be addressed, will largely determine what to simulate. As can be seen in Figure 1.2, many aspects of what to simulate are inter-related and driven based on the model purpose.

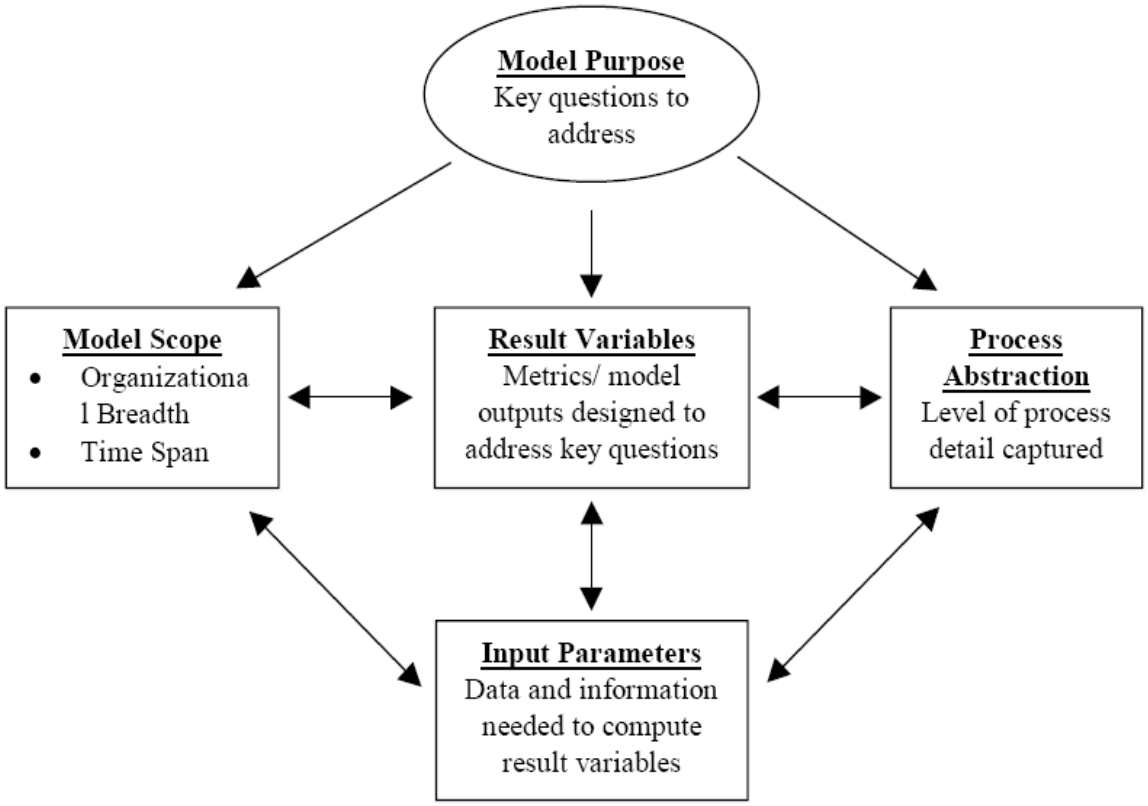

Figure 2: Relationships Among 'Why' and 'What' Aspects [56]

The following subsections would discuss aspects of what to simulate in terms of model scope, result variables, process abstraction, and input parameters, respectively in more detail.

1.1.2.1 Model Scope

Determining the model scope is an important issue and an iterative process. The scope of the simulation model needs to be large enough to fully address the main questions posed. This often requires an understanding also of the implications of the questions being addressed. For instance, a change in an early part of the software development process may cause impact also in later parts of the process, which must be taken into account when assessing the overall project performance. Therefore, the scope may vary from a small part of a software development process, such as requirements elicitation, to covering the complete process. The scope of the model needs to reflect these expanded impacts of the process change.

Moreover, in order to address the key questions related to “overall project performance”, result variables are needed to be identified to predict specific performance dimensions at the required levels. Consideration of these result variables may in turn lead to reconsideration of the model scope needed, and so on in an iterative fashion. The scope of a software process simulation is generally one of the following:

i) a portion of the life cycle (e.g., design phase, code inspection, some or all of testing, requirements management)

ii) a development project (i.e., single product development life cycle)

iii) multiple, concurrent projects (e.g., across a department or division)

iv) long-term product evolution (i.e., multiple, successive releases of a single product)

v) long-term organization (e.g., strategic organizational considerations spanning successive releases of multiple products over a substantial time period)

Large simulation studies may even include long-term software evolution covering several releases of a software product, or long-term organizational development. Thus, the scope may vary in at least three dimensions: the time span, which may vary from months to years, the entity being developed, which may vary from parts of a product to multiple products, and the organizational breadth, which may vary from parts of a team of developers to multiple teams of developers.

The time span with some approximate guidelines can be thought of as: short (less than 12 months), medium (between 12 - 24 months; about the duration of one development project, or maintenance cycle), or long (more than 24 months). The entity being developed can be a small part of a product or can be a complex system containing multiple products. The organizational breadth pertains to less than one product / project team, one product or a project team, or multiple product / project teams. It can be thus concluded that the question and its context must be well understood by the model developer in order to determine the appropriate scope for the simulation.

1.1.2.2 Result Variables

The result variables should hold the information needed to answer the key questions of the simulation study. These variables, which are heavily dependent of the key questions, could be of many different types; however, Kellner et al. [56] have listed the following most common result variables for software development process simulation models:

a. Effort/cost
b. Cycle-time (a.k.a. duration, schedule, time to market, interval)
c. Defect level
d. Staffing requirements over time
e. Staff utilization rate
f. Cost / benefit, return on investment, or other economic measures
g. Throughput / productivity
h. Queue length (backlogs)

Once again, the questions and issues being addressed largely determine the choice of result variables. For example, technology adoption questions often suggest an economic measure such as return on investment (ROI); operational management issues often focus on the traditional project management concerns of effort / cost, cycle-time, and defect levels. It should also be noted that it is quite common to produce multiple result variables from a single simulation model. When defining the result variables for a model, it is important to specify their scope. For example, are managers interested in

predictions of overall end-of-project effort? If so, the scope of the model may need to include the full process life cycle. At a minimum, the model needs to capture all process steps after the point that changes are made. This may cause the scope of the model to be modified if it has already been defined.

In most cases, as the purpose and key questions of the model are refined, model scope and result variables get refined in an iterative process. In addition, some result variables are best looked at as continuous functions (e.g., staffing requirements), and others only make sense at the end of the process being simulated (e.g., return on investment). Definition of the result variables has implications for the level of abstraction of the model and model input parameters as well. For example, in addition to predicting end-of-project effort, suppose management was also interested in effort predictions for each of the major life-cycle steps. Perhaps management is interested in detailed effort estimates for process steps within the major lifecycle steps. Requesting these result variables has significant implications for the level of abstraction of the model as well as the kinds of data and input parameters that need to be developed [40].

1.1.2.3 Process Abstraction

When the purpose, scope and result variables have been selected, the simulation modeler is confronted with selecting process abstractions and input parameters. The modeler identifies the key elements of the process, which are believed to significantly affect the result variables, and are relevant to the purpose of the model. Examples of such elements are: the key activities and tasks, the primary objects (e.g., code units, designs,

and problem reports), and the vital resources (e.g., staff and hardware). Then the relationships between the identified elements need to be described, e.g., activity dependencies, flows of objects among activities, iteration loops, feedback loops, decision points, and other structural inter-dependencies (e.g., the effort to revise a code unit after an inspection, as a function of the number of defects found during the inspection) [56].

1.1.2.4 Input Parameters

The input parameters (key factors, drivers, independent variables) to include in the model largely depend upon the result variables desired and the process abstractions identified. Typically, many parameters are needed to drive software process simulation models; some require a few hundred parameters [1]. Examples of some typical input parameters are provided below:

- amount of incoming work (often measured in terms of software size (LOC) or function points, etc.)
- effort for design as a function of size
- defect detection efficiency during testing or inspections
- effort for code rework as a function of size and number of defects identified for correction
- defect removal and injection rates during code rework
- decision point outcomes; number of rework cycles
- hiring rate; staff turnover rate
- personnel capability and motivation, over time

- amount and effect of training provided
- resource constraints
- frequency of product version releases

Some of these are usually treated as constant over the course of a simulation (e.g., the functions for effort), while others are often treated as varying over time (e.g., personnel capability and motivation). It is also noteworthy that inter-dependencies among some of these parameters may be represented in the model, rather than treating them as independent variables. For example, personnel capability may be influenced by staff turnover and hiring rates.

1.2. Problem Statement

The primary research objectives are as follows:

- To see if PSP data can be used to simulate software development.
- To create a tool that simulates the agent's activity in its organization. This means simulation at the individual level within a software development team. This research has 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 as well as organizational results. The software also validates that the PSP data can be used to simulate the software development.

- One of the goals of this research is to see if agent oriented approach can be used to simulate the software development processes. The aim was to give the agents almost same level of independence as an actual individual programmer, each agent working on his own thread, making his own decisions and using its own PSP data, on the basis of which its behavior in the team is simulated. This should help us to make a better forecast of the team's performance and therefore overall organizational performance.
- Through this research the software developer will be able to learn about software processes especially XP, through the simulator itself.

CHAPTER 2: LITERATURE REVIEW

This chapter discusses the work done in the field of software process simulation, along with the current trends, and the advantages and disadvantages of some of the earlier work done in this area. The target application areas of process simulation explored by some researchers in this field are also discussed.

2. Related Work

The ever growing need for improved strategic and operational management of software development projects is making software process simulation a popular tool. The relevance of process simulation can be placed in context by examining the approaches and techniques used to simulate the software development process in the last several years.

2.1 Simulation Approaches / Languages

Simulations have used techniques from various disciplines and the scope of work has varied from small portions of the product lifecycle to long term organizational matters. Despite best intentions, the implementation approach often influences what is being modeled. Therefore, the most appropriate approach to use will be the one best suited to the particular case at hand, e.g., the purpose, questions, scope, result variables

desired, etc. A variety of simulation approaches and languages have been applied to software processes. Some of the techniques that are more often used are discussed in detail like state-based simulation, discrete-event simulation, knowledge-based simulation, etc; other techniques that have also been used are rule-based languages [23] and Petri-net models [58].

2.1.1 State - based simulation

State-based models are based on the idea of state charts, and directly represent process activities and process dynamics through state transitions triggered by events. Design completion, for example, may trigger both the start of the coding activity and the start of unit test plan development. A state-based model easily represents parallel activities, which makes it attractive for examining questions about process bottlenecks. Raffo used a state-based simulation to predict model time, cost and quality in a detailed applied model [82]. In another paper, Raffo et al. have applied a state-based approach from systems analysis and design to the evaluation of possible process changes [83]. Since model time advances only when events occur, updates to dynamic project factors must be tied to the occurrence of events. This limits the ability of the state-based models to effectively represent feedback loops.

Raffo and Kellner describe a field study conducted at a leading software development firm using state-based simulation to quantitatively analyze the impact of one particular process change on the firm's current development process [84]. In another paper, Kellner also uses state-based simulation for a similar problem [64].

2.1.2 Discrete - event simulation

Discrete event simulation models represent the development process as a series of entities flowing through a sequence of activities. Discrete event simulation models can easily represent queues and the delays in the processing of entities that occur when resources are not available. Each entity may be described by unique attributes. Changes to the attributes as activities occur can be used to model state changes. The effort or duration of each activity may be sampled from random distributions to allow the model to represent the process uncertainty. This allows the simulation to capture the effects of variation in the entities (such as size, complexity, number of defects, defect type and so forth) on each activity.

Discrete event simulation allows us to dynamically analyze different samples of parameter values in software process instances. This enables simulated processes to function like transportation networks whose volumetric flow, traffic density, and congestion bottlenecks can be assessed according to alternative (heuristic or statistical) arrival rates and service intervals. Using discrete event simulation, process experts [22, 37, 64, 83] often find it easy to observe or discover process bottlenecks and optimization alternatives. Similarly, when repetitive, high frequency processes are being studied then discrete event simulation provides a basis for assessing and validating the replicability of the simulated process to actual experience. Likewise, discrete event simulation can be used when data on events and process step completion times/costs can be empirically measured or captured, then entered as process instance values for simulation. Thus, discrete event simulation seems well-suited for use when studying the behavioral

dynamics of processes with a large number of recurring process instances, and those with relatively short process completion (or cycle) times.

Discrete event simulation models can capture the inter-dependence that occurs between activities in a project, such as development being delayed when a programmer is diverted to another task, or testing being delayed until a test bed is released. If a model can capture these dependencies at a sufficiently detailed level, it may show ways to alter the process to reduce risk or increase productivity.

Many commercially available discrete event simulation packages now support animated visual displays of simulated process executions or results. The animated displays are used so that process experts can further validate as-is and to-be process simulations under different scenarios. These can be viewed as software development or business process "movies" [90]. In addition, the animated process instance simulations can be modified, re-executed, and viewed like other simulations.

A number of discrete event simulation models have been developed in the software process domain. Raffo et al.[83] and Host et al.[37] are to name a few. Raffo et al. embed a discrete event model in a continuous framework to understand the consequences of omitting unit tests when developers are experienced [64]. Donzelli and Iazeolla propose a two-tier approach, with a discrete event queuing network at the higher level and a mix of analytical and continuous methods at the lower [22].

Since discrete event simulation models advance time only when events occur, aspects of the system that are typically considered to vary continuously, such as schedule

pressure and productivity, can only be updated when specific events occur. This approximation is often problematic, especially when complex feedback is present [38].

2.1.3 System dynamics

The system dynamics approach to modeling complex systems was first introduced in the late 1950's at M.I.T.'s Sloan School of Management [60]. The first widely recognized application of this modeling approach to software development processes was presented by Abdel-Hamid et al. [1]. While many other modeling approaches are applicable to modeling software development processes; however, system dynamics modeling provides the ability to capture all relevant attributes of the software development process, product, and personnel which relate to the planning, tracking, and control involved in software project management. Guidelines for creating and validating system dynamics models can be found in Tvedt [98] and Richardson [88].

Lehman, Ramil and others have studied long term product evolution using a system dynamics approach [60, 87, 101] based on Jay Forrester's work on the study of social systems [29]. Abdel-Hamid and Madnick have applied the system dynamics method to study manpower and quality related issues [1]. A number of researchers have based their work on this. Pfahl and Lebsanft have used an extended model to study planning and control at the project level; several papers at ProSim 2003 applied a system dynamics approach to study elements of the lifecycle [28, 73]. In [66], Merrill and Collofello describe a process for utilizing the system dynamics model to create simulation environments suitable for addressing specific education objectives. Some

other work using system dynamics includes Stallinger et al. [95], Kahen et al. [52] and Burke [11].

2.1.4 Knowledge -based simulation

Knowledge based simulation is another kind of technology used to simulate software processes. Many approaches to modeling and simulating complex processes center on the execution of an abstract finite state machine that traverses explicitly or implicitly modeled states and/or events in the simulated process instance. For example, the Entity-Process approach developed by Kellner [50] is one that uses the state-charts tool to explicitly model and traverse the states declared in a software process.

Simulations also typically apply to instances of a modeled process, where the model is developed, then run through the simulation after its parameter values are instantiated. In a knowledge based simulation, software process states can either be explicitly declared or implicitly represented. Implicit representation means that process states will correspond to values that populate a snapshot of the underlying knowledge base of inter-linked objects, attributes, and relations used to record and manage the simulation. Knowledge based simulation can also symbolically evaluate a class of process models, or a process model without instance values. Finally, knowledge based simulation employs computational reasoning in the form of pattern-directed inferencing that is implemented via a production rule base [67]. These rules monitor the values of objects, attributes, or relations stored in the knowledge base. The set of all values in the knowledge base constitutes the implicit state of a process at a given moment. When one

or more rules is enabled and fires, then the computational procedure in the rule's action is performed, and the state of the process knowledge based is updated. Thus, these features help to begin to differentiate knowledge based simulation approaches to software process simulation.

Knowledge based simulation is useful to address different types of simulated process execution behavior. There are four types of process behavior of interest [68]. First, to understand software processes requiring fine granularity. Second, to analyze processes with multiple or mixed levels of process granularity. Granularity here refers to the level of detail that we seek to simulate in order to understand gross or fine-level process dynamics or details. Third, analyze patterns of interaction and workflow among software developers. Fourth to analyze processes whose structure and control flow are dynamically changing.

Scacchi [90], Drappa and Ludewig [23] and Storrle [96] have implemented a knowledge-based approach. Scacchi [90] described an approach and experiences in developing and applying simulation and modeling technologies to software processes.

In the work of Drappa et al. [23] the basic idea is to create a model of the software development process that can be interpreted by a simulator. The student using the simulator can control the simulated project interactively, leading it more or less successfully.

Storrle [96] has examined the properties and success factors of agile and more traditional software processes, and has also explored possibilities of blending them. He proposes process patterns as a descriptive framework capable of subsuming both

traditional and agile approaches, thus being a first step towards making agile processes scalable.

2.1.5 Activity – based simulation

Although many different simulations techniques have been used, all applications that have been found so far focus on activities or phases rather than individuals. The software development process is modeled as a set of activities implicitly performed by developers in which some artifacts are produced in support of subsequent activities. Alternatively, the activities have people as input and output, for instance hiring or training staff [11]. What activities and in which order they are performed are generally specified in a static and 'ideal' process model such as the analysis process shown in figure 2.3. Note that in a static process model activities are related to some 'role', (shown here in the upper left corner of each row), in the software development process, but this relationship is generally not taken into account in the simulation models.

In activity-based approaches, the software developers are described in terms of averages of all individual developers in the organization. For instance, the time taken to complete an activity is usually determined using an average productivity rate of all developers in the organization. Alternatively, the individuals are divided into a number of categories, and the number of individuals in each category is used to describe the state of the organization [102].

One explanatory and representative example is reported by Burke [11], where the staff members were categorized as being for or against process improvements, and

experienced or recently hired respectively. Rather than assigning these characteristics to individuals, the numbers of individuals belonging to each of the categories were modeled. In effect, the modelers are assuming homogeneity among the individuals, whereas the individuals in actual software development organizations are highly heterogeneous. A result from using system level variables to drive the simulation is that local variations in an organization cannot be predicted or explained. For example, Burke reported that one development team in the organization he modeled was far more efficient and more in favor of process improvement than other teams in the organization, something that could not be explained nor predicted in his model.

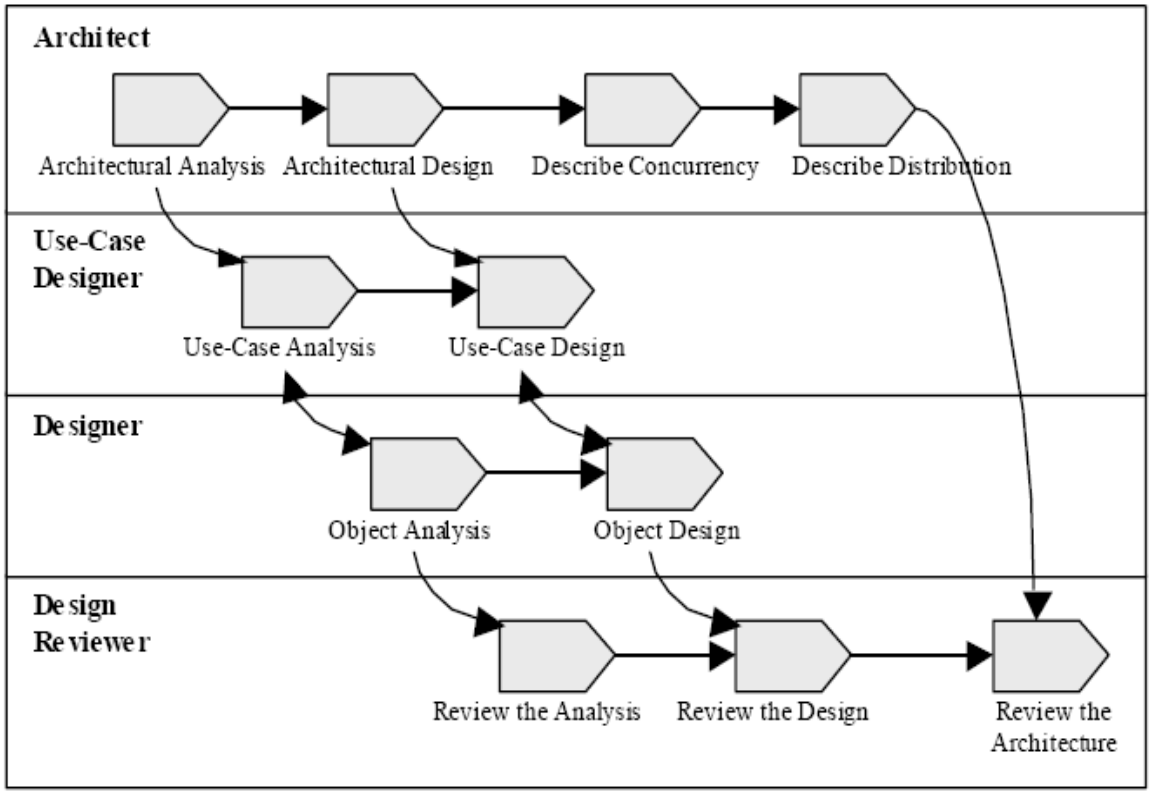

Figure 2.3: Example of an activity-based software development process model [102]

Parunak et al. [76] have presented several examples where false assumptions of homogeneity in simulation models have lead to problems. In many software development process simulations, the software development process is modeled as a flow of activities according to a predefined process and decisions are not explicitly modeled at all. Other software development process simulations, such as most system dynamics simulations, attempt to model management policies [102]. However, these simulations suffer from some of the problems that arise when discrete decision-making is modeled using equation-based simulations, which have been discussed by Parunak et al. [76]. Kellner et al. [56] suggest that the most appropriate simulation modeling technique should be selected based on scope, result variables desired, etc. However, which approach is better suited for a certain situation is a question that has only been scarcely researched. Some guidance is given regarding the dimension of time. Kellner et al. also state that system dynamics tend to be more convenient for analyses above the detailed process level, whereas discrete event and state-based simulations tend to be more convenient for detailed process analyses.

2.1.6 Object – oriented simulation

The object-oriented paradigm is a development strategy based on the concept that systems should be built from a collection of reusable components called objects. Instead of separating data and functionality as is done in the structured paradigm, objects encompass both [51]. Object orientation systems view the world as a set of autonomous agents that interact or work together to solve some complex task. Each object is

responsible for a specific task that helps one organize the complexity of complex systems which simplifies the computer programming tasks. Object-oriented designs yield smaller systems through the reuse of common mechanisms.

In the paper by Melis et al. [65], they present a simulation model that is developed to evaluate the applicability and effectiveness of Extreme Programming (XP) process. The XP process has been modeled and a simulation executive has been written, enabling to simulate XP software development activities. The model follows an object-oriented approach, and has been implemented in Smalltalk language, following an XP process itself.

2.1.7 Analytical simulation

Most of the process models used by the software community are analytical models, which provide average data on process behavior. Examples of such models are: function point-based models [3] to estimate the final product size and associated effort; the COCOMO model [10] to estimate schedule and effort for a given software product; the Rayleigh model [81] to shape the staffing profile, and the reliability growth models [27] to estimate the testing effort. Besides providing average data, these models cover only a strict subset of the process attributes, separate the effects of such attributes (e.g. the development effort from product defect density), and do not permit analysis of the process behavior in a perturbed environment (e.g. changes in product requirements, staff reductions etc).

2.1.8 Multi-agent based simulation

Multi-agent based simulation differs from other kinds of computer-based simulation. In multi-agent based simulation, an organization can be viewed as a collection of agents, interaction possibilities and informational artifacts. By agents, we mean individuals or possibly teams with some set of core skills or competences and some ability to learn and adapt. The relationships between the agents determine the interaction possibilities – for example, manager agents may co-ordinate with other manager agents and delegate to lower level operative agents. As multi-agent based simulation, and other micro simulation techniques, explicitly attempts to model specific behaviors of specific individuals, it may be contrasted to macro simulation techniques that are typically based on mathematical models where the characteristics of a population are averaged together and the model attempts to simulate changes in these averaged characteristics for the whole population. Thus, in macro simulations, the set of individuals is viewed as a structure that can be characterized by a number of variables, whereas in micro simulations the structure is viewed as emergent from the interactions between the individuals.

Activity-based simulation is a more natural choice when the behavior of the individuals is known only in terms of a predefined flow of activities and it is reasonable to assume that the defined process is followed. Other important assumptions are that localization of information and characteristics of individuals do not significantly affect the outcome of the process. Note that the applicability of an individual based approach is not limited to a certain organizational scope. Rather, the scope of the simulation affects

which abstraction level the individuals are conveniently selected, and on which abstraction level the behavior of the individuals are most conveniently modeled.

Multi-agent based simulation is likely to be a more appropriate tool for software development process simulation than activity-based simulation techniques when the outcome of the software development process under study is determined by discrete decisions made by individuals interacting with each other and their environment. Multi-agent based simulation can also be useful when the need is to simulate a specific organization or project where the characteristics of the individual software developers are known, or to study the sensitivity of the software development process to individual characteristics, e.g., measure the impact of changes in behavior of individuals. When there is a high degree of locality either in terms of characteristics of the environment, the individuals' characteristics, or of the availability of information, multi-agent based simulation might be useful. Also, management policies and other descriptions of individual behavior can be captured in the model with the use of multi-agent simulation [102].

2.2 Simulation Techniques

The purpose of this section is to highlight some of the important capabilities and techniques that should be considered when selecting a particular simulation tool. This set of considerations is based upon the extensive literature study done on the tools used by experts for developing software process simulations in industrial settings.

A model may be depicted in a form that is primarily visual or textual. Visual models (i.e., graphical, diagrammatic, or iconic) have become the norm for software process simulations because they promote understandability and ease of development. These tools often include an ability to animate the model during simulation to show the flows of objects (e.g., code units, designs, problem reports) through the process, the activities currently being performed, and so forth. This can be very helpful as a further aid to understanding, and as a tool to use during validation of the model with process experts. Even when a model is primarily visual, it almost invariably entails supplemental textual information specifying inter-relationships among components, equations, distributions for random variables, etc. As a result, suitable repository capabilities that can store, present, and report this information easily is highly desirable [56].

Another useful capability offered by many simulation tools is the simultaneous reporting of results (usually plots of result variables over time) while the model is executing [102]. Many tools support interactive simulation, where the user specifies or modifies some of the input parameters during model execution rather than only beforehand, and/or can step the model through its execution. Some tools also allow batches of simulations to be executed automatically from a single set-up, and results accumulated across the individual runs in the batch.

A simulation can be deterministic, stochastic, or mixed. In the deterministic case, input parameters are specified as single values (e.g., coding for this unit will require 5 work-days of effort, or 4 hours per hundred lines of code; there will be two rework cycles on code and unit test; etc.). Stochastic modeling [40] recognizes the inherent uncertainty

in many parameters and relationships. Rather than using (deterministic) point estimates, stochastic variables are random numbers drawn from a specified probability distribution. Mixed modeling employs both deterministic and stochastic parameters. In a purely deterministic model, only one simulation run is needed for a given set of parameters. However, with stochastic or mixed modeling the result variables differ from one run to another because the random numbers actually drawn differ from run to run. In this case the result variables are best analyzed statistically (e.g., mean, standard deviation, distribution form) across a batch of simulation runs; this is termed Monte Carlo simulation [42]. Although many process simulation tools support stochastic models, only some of them conveniently support Monte Carlo simulation by handling batches well.

Finally, sensitivity analysis is a very useful technique involving simulation models. Sensitivity analyses explore the effects on the key result variables, of varying selected parameters over a plausible range of values. This allows the modeler to determine the likely range of results due to uncertainties in key parameters. It also allows the modeler to identify which parameters have the most significant effects on results, suggesting that those be measured and/or controlled more carefully. As a simple example, if a 10% increase in parameter A leads to a 30% change in a key result variable, while a 10% increase in parameter B leads to only a 5% change in that result variable, one should be somewhat more careful in specifying or controlling parameter A. Sensitivity analysis is applicable to all types of simulation modeling. However, it is usually performed by varying the parameters manually, since there are few simulation tools that automate sensitivity analyses [39].

2.3 Data / Measurement Issues

Simulation models should undergo calibration and validation to the extent possible. Validation can be performed, in part, by inspections and walkthroughs of the model (providing what is often referred to as face validity). In addition, actual data should be used to validate the model empirically and to calibrate it against real-world results. Considerations of the questions to be addressed, desired result variables, input parameters, validation, and calibration, often suggest metric data (measurements) that would be valuable to collect, and show how they are useful. Unfortunately, a lack of relevant, desired data in practical settings is all too common [56]. Accurate simulation results depend on accurate values of the parameters. Similarly, calibration also depends on accurate measurement of the results seen in practice. In many "real world" settings these variables have not been measured, or at least not carefully measured. Strategies that can be useful in coping with this situation include the following [53]:

i) Adjust existing values to approximate desired variables, e.g., if cost is desired but only effort is available, one can generally approximate cost using an average hourly rate.

ii) Construct values from other detailed records, e.g., data on when defects were injected (by phase) may not be available as part of summary reports from inspections, but might be found on the original, detailed, inspection forms.

iii) Obtain estimates from personnel involved, based on their past experience or hypothetical expectations when necessary.

iv) Utilize typical values taken from the literature, e.g., defect detection efficiencies for source code inspections and unit tests, measured in other organizations, are widely available.

2.4. Application Areas in Software Engineering

Simulation has been applied in many fields, such as aerospace and energy production, but to date, it has not seen broad practical application in software engineering. This may be because it is more difficult to accurately model human and organizational behavior than to model physical systems, or it may be that the emphasis on software process is a relatively recent phenomenon. Whatever the reason is, it is unfortunate because the rewards from its use are myriad. In this section, applications of simulation are briefly reviewed and some of the benefits that can be obtained are discussed.

2.4.1 Requirements Management

Simulation can be extremely helpful in pinning down non-functional software system requirements early in the product lifecycle, particularly when examining temporal behavior. Simulation can mimic the performance characteristics of software components and their interactions, the effects of time delays and feedbacks, and of finite capacities and resource bottlenecks [4, 61]. Requirements are rarely static but evolve as experience grows with product development. Thus, simulation is not only a valuable tool in defining the initial requirements but also can be used to test alternate modifications prior to their

implementation. Finally, a system simulation can be viewed as a component of the requirements and can provide quantitative measures against which the target software system must comply. The processes through which the requirements are managed are also critical [15]. However, as far as modeling is concerned, such processes have much in common with other project management processes, e.g., design, development, and test. Thus, the discussion in the next section is relevant to requirements management.

2.4.2 Project Management

Simulation can allow managers to make more accurate predictions about both the schedule and the accumulated costs associated with a project [1, 55]. This approach is inherently more accurate than costing models based on fits to historical data because it accounts for the dynamics of the specific process. With regard to schedule, simulation can account for dependencies between tasks, finite capacity resources, and delays resulting from probable rework loops. Some simulation tools also allow one to compute the accumulation of costs on an activity-dependent basis. These features are useful for generating proposals that are more accurate in cost and schedule and therefore more likely to keep a company in business.

Software project management skills are becoming an important component of software engineering education. Software engineers working in teams need to carefully plan and coordinate their efforts in order to be successful. Unfortunately, most universities provide inadequate education in software project management. Most of the universities use lecture-based approaches which provide the necessary steps in software

project management, but are deficient in providing the students with hands-on experience. Software simulation provides a bridge between course-based and hands-on experience. It provides an interactive environment of repeatable exercises. It also provides a medium for measurable evaluation of student performance which can be used to customize the education process to fit the needs of individual students. The work of Merrill et al. [66] focuses on using system dynamics modeling for simulating software development activities because of its ability to dynamically represent relevant project attributes in the software development process. They successfully developed and validated a system dynamics model of the incremental software development process. They describe a process for utilizing this system dynamics model to create simulation environments suitable for addressing specific education objectives, along with the discussion on the benefits and guidelines for using tools of this kind.

2.4.3 Training

Because of the complex dependencies between attributes of organizational systems, these systems can respond in counter-intuitive ways. The classic example is Brooks' law [31], which states that hiring people late in a project can further delay the project. Simulation can play an important role in sensitizing managers to the consequences of instabilities that result from the system feedbacks often inherent in badly designed organizational processes. Simulation-based training can also provide the software development managers with the insights necessary to establish effective processes and to operate these processes in a stable manner. Thus, the focus is to train

management in the design and operation of software processes, not in the technologies that support software development. Simulation-based training can be performed by individual managers who interact with the simulated software development activities. Generally, the user has control over certain control parameters (such as hiring rate and salaries), and the decisions made alter the course of the subsequent simulation history. Analysis of the training session can be performed after the session to see what went right or wrong in the decision making process and to reinforce effective decision making. To bring groups of managers to a central location for training can be both costly and time consuming. In the near future, such groups may be trained in a geographically distributed manner using a simulator with displays on the managers' local terminals. Interaction between trainees allows for joint decision making. This interaction can be provided through the use of collaboration technology. With the increasing interest in this technology distributed training may soon become practical [15].

Training and teaching is an important application of software process simulation [56]. Up until this point, however, it has only been used in the context of students running simulations of process models that were built by someone else. Various models and environments have been developed targeting this context, e.g. [5, 20, 24, 71, 80, 94]. These all share the purpose of giving students virtual experiences of realistic software processes that would otherwise be infeasible to practice in an academic environment. So far, the reported usage of simulation and modeling in this context is always structurally similar: An existing model is used by the trainee for virtual experiences, i.e. simulation is

leveraged for teaching, and modeling is done outside the learning situation by an instructor or some other expert beforehand.

Simulation/adventure games, such as The Sims [26] and SimCity [25], provide a tremendous source of experience and technology that can successfully be adapted to illustrate the software process. In these games, players strive to fulfill certain—sometimes conflicting—goals by living "virtual lives" in an environment where they are forced to make constant decisions. It is interesting to observe that, in experiencing the consequences of their decisions, players implicitly undergo an experience similar to the software process.

Although software engineering process education is one domain in which simulation would be an ideal educational tool, the field has yet to fully leverage this approach. There have been a few exceptions that have identified promising avenues as in the work of [20, 24, 75, 79, 94], but these have not yet fully pushed the boundaries of simulation in software engineering education. Mainly, these approaches have been limited in one or more of the following areas: interactivity, customizability, and/or graphics.

To address these issues, Navarro et al. [71] developed SimSE, an interactive, graphical, educational software engineering simulation game designed to teach students the process of software engineering [72]. SimSE addresses the large gap that exists in traditional software engineering educational techniques – students are exposed to several software engineering concepts and theories in lectures, but have limited opportunity to put these ideas into practice in a typically small software engineering project. SimSE

aims to fill this gap by providing students with virtual experiences of participating in quasi-realistic, large-scale software engineering processes. SimSE is a single-player game in which the player takes on the role of project manager of a team of developers.

The educational, interactive, and graphical nature of SimSE imposes three unique requirements upon its process modeling language: First, it must be predictive, i.e. it allows the modeler to specify causal effects that the player's actions will have on the simulation. Second, it must be interactive, meaning that it should operate on a step-by-step basis, accepting user input and providing feedback constantly throughout the simulation. Finally, it must allow the modeler to specify the graphical representations of the elements in the model.

Existing process modeling approaches are mostly predictive [1, 7, 59] or prescriptive [12, 74], but not both. Few are interactive; few support graphics [38, 63]; and none fulfill all of these requirements. The closest fit is the modeling language used in SESAM [72], another educational software engineering simulation environment. However, despite the fact that the SESAM language is highly flexible and expressive, the model-building process is learning and labor intensive and requires writing code in a text editor. Furthermore, the user interface for the simulation is text based, so the modeling language has no support for graphics.

Birkhoelzer et al. [5] suggest a different approach: to use the modeling activity for teaching as well, rather than the simulation activity only. In particular, they propose to assign students the task of building a new software process simulation model using the above mentioned educational software process simulation environment SimSE.

2.4.4 Process Improvement

Simulation can be used to support process improvement at all levels of the Capability Maturity Model (CMM) but particularly at the higher levels [34, 86, 99]. Because simulation forces one to address metrics and process behavior in a methodical way, one may argue that simulation can accelerate the introduction of process improvement. Consistent with the philosophy of the CMM, simulation capability at each CMM level incrementally builds on the simulation capabilities of the preceding levels and matches the needs of the software engineering practices at that level [15]. Traditionally, revised or new processes are improved through operational experience. This can be expensive and risky. Simulation can provide considerable insights into how a process will work prior to its implementation. These insights can help the process designer assess alternatives and show that a specific process design performs in a manner that meets expectations. In this way, processes can be pre-tested, and buy-in is more likely obtained from management. Subjective criticisms are less likely, since quantitative simulation of validated models can produce specific and credible answers to perhaps hostile questions.

Ruiz et al. [89] present an integrated framework for software process improvement according to the Capability Maturity Model (CMM). The framework is double-integrated. First, it is based on the systematic integration of dynamic modules to build dynamic models that model and simulate each maturity level proposed in the reference model. As a consequence, a hierarchical set of dynamic models is developed following the same hierarchy of levels suggested in the CMM. Second, the dynamic

models of the framework are integrated with different static techniques commonly used in planning, control, and process evaluation. The paper describes the reasons found to follow this approach, the integration process of models and techniques, the implementation of the framework, and shows an example of how it can be used in a software process improvement concerning the cost of software quality.

2.4.5 Architecture and Commercial-Off-the- Shelf Integration

Building complex software systems usually begins with addressing the system's architecture. Without a firm notion of how the major components of a software system interact, there is little likelihood that the system will reflect performance effectively. One would like to know early in the development cycle that such attributes as reliability, reusability, maintainability, portability, performance, and modifiability are above some acceptable level. There are complex dependencies between these attributes. For example, in improving performance, reusability might be sacrificed; or in improving portability, maintainability might require increased effort. Making trade-offs in this multidimensional space is not easy, but if they are not made at a high level of design abstraction, there is little chance they can be dealt with once coding begins. Simulation is a tool that can be used to examine some of these architectural trade-off issues [46]. Simulation can provide early insights into timing, resource usage, bottlenecking, and usability. In addition, one can rapidly gain insight into the implications of design changes by running simulations with varying independent parameters. Finally, one can assess sensitivities to parameter changes in a Monte Carlo (statistical) sense.

2.4.6 Product-Line Practices

Simulation makes considerable sense in the economic analysis of product lines. In particular, “Because product-line development involves changes in product composition and production, software size measures, such as lines of code, are not good predictors of productivity improvements. To estimate, track, and compare total costs of disparate assets, adaptation of other cost modeling techniques, particularly activity-based costing to asset based software production, is needed.” [45]

A software product line is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission, and are developed from a common set of core assets in a prescribed way. A software product line approach promises shorter time-to-market and decreased life cycle cost. However, those benefits are not guaranteed under every situation and are affected by many factors, such as number of available employees, market demands, reuse rate, process maturity, and product line adoption and evolution approaches. Before initiating a software product line, an organization needs to evaluate available process options in order to see which one best fits its goals [14].

Software product line development involves three essential activities: core asset development, product development, and management. Core asset development (domain engineering) involves the creation of common assets and the evolution of the assets in response to product feedback, new market needs, etc. Product development (application engineering) creates individual products by reusing the common assets, gives feedback to core asset development, and evolves the products. Management includes technical and

organizational management, where technical management is responsible for requirement control and the coordination between core asset and product development.

A software product line can be initiated under several situations: independent, project-integration, reengineering-driven, and leveraged [93]. Under the independent situation, a product line is created without any pre-existing products. Under the project-integration situation, a product line is created to support both existing and future products. Under a reengineering-driven scenario, a product line is created by reengineering existing legacy systems. And the leveraged situation is where a new product line is created based on some existing product lines. Several software product line cost estimation approaches [6, 9, 18] have been proposed.

Bockle et al. discusses software product line adoption scenarios and presented a product line cost model [6]. Among the seven adoption scenarios, only two of them, developing a single product line without pre-existing products, are considered in their paper. Their cost model takes organization costs into account, which is not considered in their work.

Boehm et al. proposes COPLIMO [9], a COCOMO II [8] based model, for software product line cost estimation. COPLIMO has a basic life cycle model, which consists of a product line development cost model and an annualized post development extension. Their simulator uses COPLIMO as the underlying cost model. However, in the implementation, the cost model is designed as a plug-in model, thus allows other cost models to be used as well. The early stage simulator [14] used the COPLIMO basic life

cycle model. To allow more detailed modeling, the basic life cycle model was extended by using COCOMO II [8] and was used by the architecture-based simulator [6].

The paper by Chen et al. [13] aims at developing a software product line process simulator that can predict the cost for a selected software product line process and provide useful information for cost-benefit analysis. In this work the authors describe a simulator that is intended to support post-architecture software product line process impact estimation. It uses DEVSJAVA [103] as the modeling and simulation formalism, COPLIMO [9] as the cost model, and Microsoft Project [69] as the process definition tool. The simulator can provide both dynamic and static information for the selected process. Stepping through the simulator allows tracing product line process and uncovering hidden problems. The static results generated at the end of the simulation gives time, cost, and resource estimates for the selected process. By running different processes through the simulator and comparing their results, an organization can evaluate the alternative processes.

2.4.7 Risk Management

Projects are often vulnerable to risks. These risks can result from, for example, organizational disruptions, changing staff levels, requirements ratcheting, and funding cuts. Simulation helps in identifying associated project risks early on. Simulation can help design more objective, less risk-prone strategies by quantitatively predicting the consequences of alternate decisions. Risks are also associated with alternate system architectures or commercial off-the-shelf integration strategies. The pros and cons of

different design decisions can be identified by using simulation to examine the potentially complex interactions of alternate component configurations [56].

2.4.8 Acquisition Management

Acquisition management is dependent on the above mentioned areas, such as requirements management, project management, and risk management. As these areas can benefit a lot from the use of simulation, so can acquisition management. Simulation is used for the validation of a contractor's estimates of costs and schedules. Also, it provides insights into the contractor's design ability to meet system requirement. The potential contractor problems can be predicted via the use of simulation before they take the shape of reality. It also has the effect of keeping the contractor honest in its estimates of cost and schedule [56].

Scacchi and Boehm [41], [92], have addressed the subject of simulation-supported acquisition in some detail. They address the issues of how simulation can support the acquisition lifecycle. According to them, the three simple reasons for using simulation in software acquisition management are: Firstly, it helps in the early identification and reduction of the risk associated with complex software acquisition. Secondly, it helps in understanding what kinds of system requirements and architectures would be feasible and affordable as per the various programmatic and technological constraints. Thirdly, simulation can be used to gain insights into how to better manage the system engineering effort so as to improve the overall likelihood of a successful acquisition effort. Modeling and simulation can be used to help identify where consensus can be established and

validated, as well as to identify where disagreements can be found, so their consequences can be examined. They also give specific examples of potential applications, and suggest that a research and development effort be established to explore issues such as virtual prototyping, incremental iterative acquisition supported by simulation, and the use of wide-area collaborators.

CHAPTER 3: RESEARCH DESCRIPTION

This chapter describes the solution to the problem stated in chapter 1 of this dissertation. It discusses the uniqueness of the proposed solution and the benefits for the industry through this work. This chapter also gives the details of the methods and techniques used in this work along with the discussion on the uses of this system.

3.1. Introduction

To whet their freshly learned project management skills, software engineering students need to practice software development techniques that are not just academically relevant but also used by the industry. PSP [48, 49] and Extreme Programming [43, 47], the former a process for personal use and the latter for teams, are used frequently both for academics and industry.

Personal software process or PSP is designed with an aim to improve the quality and productivity of the personal work of individual software engineers. 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. can be determined using PSP, some of which can be used to simulate the activities of team members.

Extreme Programming or XP is designed as an agile software development process. It is a process that can be used to co-ordinate the activities of the teams, and can aid in rapid development of software projects with user satisfaction.

The Primary research objectives are as follows:

- To determine if PSP data can be used to simulate software development.
- To create a tool that simulates the agent's activity in its organization. This means simulation at the individual level within a software development team. This research simulates each individual and then assembles the results for the entire team. It models each individual using his/her production metrics obtained from the PSP process, and combines these individuals to determine the team results as well as organizational results. The software also validates that the PSP data can be used to simulate the software development.
- One of the goals of this research is to see if an agent oriented approach can be used to simulate the software development processes. The aim was to model each agent as if it were an actual individual programmer. Each agent works on its own thread and makes decisions using its own PSP data. This is intended to assist us to make a better forecast of the team's performance and therefore overall organizational performance.
- Through this research the software developer will be able to use the simulator to learn about software processes, XP in particular.

3.2. Description

This research developed a model for simulating small-team software processes from a role playing prospective. It was aimed at producing a software tool that simulates a software development project to see if PSP data can be used to simulate software development process at individual developer's level. The research was carried out using the iterative (spiral) model life cycle. This choice was made as it was very difficult to identify all the requirements for the system at the beginning of this project.

In this computer-simulated project, the user of the tool plays the role of the lead software developer. The user guides a simulated software project from start to end using a game-playing paradigm. Thus user gains first hand experience with verification, manipulation and observation of the software process.

The user of the system takes the role of the project leader, and provides the planned tasks and the team assignments for the iteration for the simulation. The user decides the number of developers to be included in the simulation along with their performance data like productivity, errors per LOC etc. This performance data is the historical data collected from PSP for each developer. The project leader also provides the expected LOC count and/or expected time required to complete all the user stories for the given project.

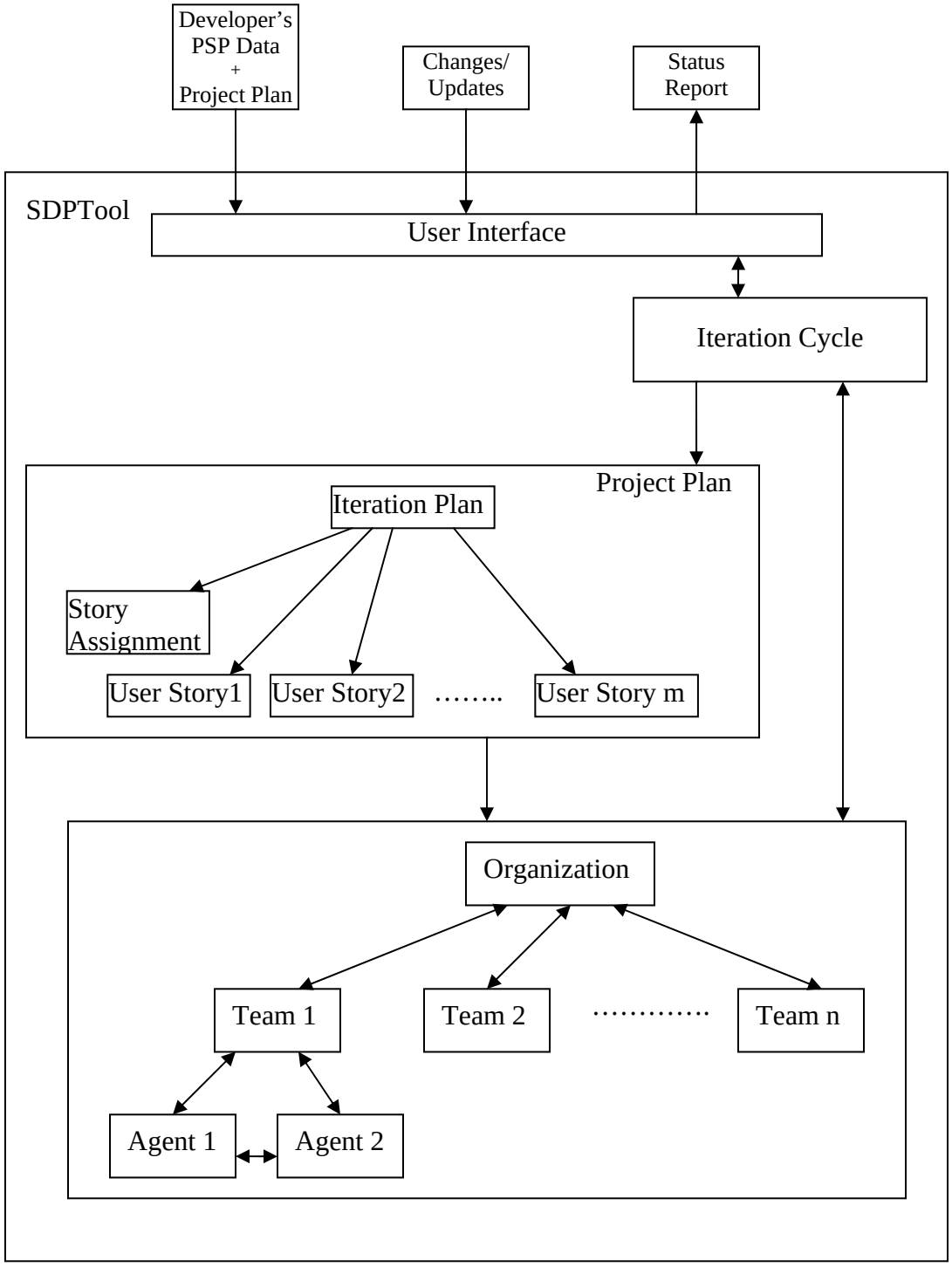

Figure 3.1: Model for Agent Based Simulation of Software Development Process

The simulation is developed in C#. It is a multi-threaded, multi-agent simulation. The agents imitate the behavior of software developers within a team. The teams are initialized by the team manager, or the user of the system. Each agent has a different behavior pattern defined by the developer it's imitating. The PSP data is used to initialize the behavior of each team member or agent. Constant feedback is provided back to the user at each interval of simulation time. The tool provides the user with the flexibility of modifying the project (stories, or features), teams and/or story development priority list.

The simulation is based on randomly generated values defined by the programmer's performance data. The input for the tool is the project development plan along with the expected values for each user story (in size and/or time). The user can also choose between different measurements such as project velocity, component status, personnel effort, earned value, productivity, component size, lines of code, and defects. In the simulation, the actual story size is estimated based on the regression parameters for each developer as shown in equation (1), and then averaged between the team members to find the actual size of the user story to be developed.

$$size = \beta_0 + estimatedSize * \beta_1 \quad \text{---------- (1)}$$

The variables, pspUpperLocPerHour and pspLowerLocPerHour are the bounds on the productivity of the agent. According to the latest version of PSP, the productivity of a developer generally ranges within these bounds and we used this assumption for this simulation as shown in equations (2) and (3).

$$newLoc = \frac{pspLowerLocPerHour + rand() * (pspUpperLocPerHour - pspLowerLocPerHour)}{60} \quad \text{---------- (2)}$$

$$defectsIntroduced = \frac{pspCodeDefects}{100} * pspDefectsPerLoc * newLoc * rand() \quad \text{-------- (3)}$$

Also, during the simulation, an agent is randomly selected from the two team members and is responsible for the design, or coding while the other reviewed his work.

Once all these inputs are given, the software development process tool (SDPTool) starts simulating the phases of a software development process and the development of the user stories. The simulation is controlled by the agent's PSP data. The results at the end of each simulation's time step are the average of all the agents' performance.

At the end of each iteration cycle, the status of the project is displayed to the user. The simulation outputs the actual duration values for each user story, the organization/team/agent productivity, and story status. The user can then compare the displayed planned results to the actual results generated by the simulation. The user has the advantage of being able to alter resources, modify tasks, or make other changes and observe their effect on the simulation. The user can also update the project requirements if he so desires, on the basis of values generated through simulation.

CHAPTER 4: APPLIED RESULTS AND RESEARCH VALIDATION

The experimental setup and the results are discussed in this chapter. It describes the methods used to collect the necessary data to execute the experiment and the results collected. There were three levels of validation. In the first level, a one-to-one comparison was done between the core practices of eXtreme Programming and SDPTool. In the second level, the simulation was validated by inspection done by extreme programming expert. At the third level of validation, the results from the simulation were compared with the real life data.

4.1. Data Collection

The data required to model each developer was LOC developed, defects removed, LOC per hour, upper LOC per hour, lower LOC per hour, defects per LOC, planning time, design time, design review time, code time, code review time, compile time, test time, refactor time, planning defects, design defects, code defects, compile defects, test defects, refactor defects, planning defects removed, design defects removed, code defects removed, compile defects removed, test defects removed, refactor defects removed, planning fix time, design fix time, code fix time, compile fix time, test fix time, and refactor fix time.

Upper loc per hour and lower loc per hour are the bounds on the productivity of the agent. These were calculated from about seven assignments that each student did as a part of their software process course.

In the PSP data that we had for the students, there was no data for refactoring. But, the values for refactoring were needed in order to simulate the refactoring practice of extreme programming. The assumption that was made in order to accomplish this was to use the values of postmortem for refactoring. The reason behind this choice was that the students initially refactor their work during postmortem and this was the closest to the refactoring data that we could obtain from the PSP data.

The data for each senior project was also needed, such as, the project development plan which included the team members, user stories, expected and actual completion time for each story, and user story allotted to each teams for each iteration cycle. This data was obtained from the documented reports that the students submitted as their final reports for the senior design class.

4.2. Results Collected

Once the required data was collected, the simulation computed the actual development time for each user story, defect rates, productivity, project velocity, and project status at the end of each iteration cycle. At the conclusion of the simulation, the data collected was compared with the actual data collected by the students for their projects.

4.3. Validation by Comparison of Features

A thorough mapping of features between eXtreme Programming and SDPTool was performed. The comparison of the twelve core practices of XP and SDPTool are discussed in Table 4.1.

Table 4.1: Twelve Core Practices

Twelve Core Practices	Extreme Programming (XP) [43, 47]	Software Development Process Tool (SDPTool)
The Planning Game	The main activity in the planning game is the writing-estimation-prioritization back and forth negotiation of stories between programmers and customers. The customer and development teams get together to decide on what features of the required system will be of maximum value to the business.	The planning game is done by the user of the system, even before he starts the simulation. The plan is the starting point of the simulation, as to where the user initializes the simulation using the project plan he/she has created. The expectations from the user are that he/she will complete writing all the stories on cards, estimating, and prioritizing them before using the system. The plan developed by the user is the initial input to the tool.
Small Releases	A simple system containing a useful set of features is put into production early and updated frequently in short cycles.	Early and often releases can be applied with SDPTool as it allows short releases and user-defined intervals. The user controls the interval length of the simulation by defining it in the project plan and also can be changed during the simulation.
Metaphor	Each project has a “system of names” and description which helps to guide the	A user of the system can use and refine whatever metaphor proves best. It is

	development process and communication between all parties.	expected that the user chooses a metaphor that helps him/her understand the parts of the system he/she is talking about.
Simple Design	The simplest design is always used to build the application as long as it meets the current business requirements. Do not worry about future requirements, as requirements change with time anyway.	The design is controlled by the user of the tool. The project leader or the user creates a project plan as discussed in the planning game and uses it to run the simulation.
Testing	Software developed with XP is validated at all times. Before new features are added tests are written to verify the software. The software is then developed to pass these tests.	When developing a new module, the user should design the interface first, then the unit test, and only then go on to implement the module. The user should be completing these tasks before inputting the values to the simulation.
Refactoring	This is a technique for improving the design of an existing codebase. Its essence is applying a series of small behaviour-preserving transformations that improve the structure of the code. By doing them in small steps you reduce the risk of introducing errors.	The refactoring phase is implemented in the simulation itself, just like other development phases. Once the story is completed in the simulation, it is refactored. The refactoring time depends on the agent's PSP data. Postmortem time is used instead of the refactoring time, as there is no concept of refactoring in PSP.
Pair Programming	Programmers using XP are paired and write all production code using a single machine per pair. This helps the code to be constantly reviewed while being written. Pair	In SDPTool, two agents are always a part of a team, and pair programming is extensively used in the simulation. When one agent is designing or coding, the other one is

	Programming has proved to produce high quality code with little or no decrease in productivity.	doing the reviews. The simulation randomly selects one member between the two team members and that member codes while the other reviews the code.
Collective Code Ownership	All the code belongs to every member of the team, no single member of the team owns a piece of code and anyone can make changes to the codebase at any time. This encourages everyone to contribute new ideas to all segments of the project.	In SDPTool, all the user stories have descriptors which collect the history for each user story, e.g. time spent, error introduced, current errors, LOC, etc. So if a different team takes over a story, it can start from where the story was left off without losing any prior information.
Continuous Integration	The aim of continuous integration is to prevent or reduce code from spreading from the main codebase; the more frequently code is integrated into the main codebase the less chances that there will be diversion.	The simulation integrates all the project history at every time interval. Once a story gets completed, the simulation integrates the project by combining the performance data of all the user stories and the teams.
40-Hour Week	Programmers in an XP project normally adhere to a 40 hour working week in order to maintain productivity and avoid burn out.	The simulation works assuming it's a continuous development process and does not consider the concepts of days and weeks. It accounts the time spent in minutes. The user can always compute the time in days or weeks if he needs to and can always adhere to a 40-hour week.
On-site Customer	One or more customers who will use the system being built are allocated to the development team. The customer helps to guide the development and is	All the communication is the responsibility of the user of the system. The simulation is not responsible for any communications with the

	empowered to prioritize and state requirements or answer any questions the developers may have. This ensures that there is effective communication with the customer and as a result less documentation will be required.	customers.
Coding Standards	Everyone use the same coding standards which makes it easy to work in pairs and share ownership of all code. One should not be able to tell who worked on what code in an XP project.	The simulation computes the matrices assuming coding standard is followed.

4.4. Validation by Inspection

The simulation was inspected by Dr. Chapman, who is the lead instructor of the senior design class in the Department of Computer Science and Software Engineering at Auburn University. He is an expert in the field as he has been teaching senior design for past twelve years and has been using eXtreme Programming in his class for the past six years. He was given a brief introduction of the simulation and the research objectives. Also, he had the access to the source code and the data required for running the simulation, along with the executables. He verified the features needed in the simulation that is critical to successful completion of the senior design projects using the version of XP they follow and was satisfied with the implementation of the features and concluded that the simulation is consistent with the features used in his senior design classes.

4.5. Statistical Validation

Validation of the tool was done by comparing actual results with the simulated results for four different senior design projects. The projects were independent of each other and each project had four members who were responsible for different features of the project. The team members' performance was initialized based on their PSP historical data (shown in Appendix B), that was collected as a part of software process class. The four members were split into two teams and the story list was divided into the teams (shown in Appendix C). For each iteration cycle, project plan was made and the members were responsible for the completion of their halves. The performance data was collected by the team members as a required part of the course and this collected data was used in the validation process of this research.

The simulation ran for 30 times for each project and the final result was the average of these runs. The final results of the simulation were then compared with the actual results collected by the project team members. The comparison of the results is shown in Table 4.2.

In order to determine the statistical validity to the results, correlation was chosen to be a good statistical measurement. The reason for this choice was that this data was independent of each other between projects as each project was different. And even within same projects all the stories were unique and thus, independent [100].

Table 4.2: Comparison of Planned, Actual and Simulation Results for all Projects

Story Name	Time (mins)			Story Name	Time (mins)		
	Planned	Actual	Simulated		Planned	Actual	Simulated
P1.1	480	660	549	P2.11a	360	360	486
P1.2	360	1410	454	P2.11b	240	240	393
P1.3	900	3060	805	P2.11c	240	240	390
P1.4	360	330	239	P2.11d	180	180	201
P1.5	300	240	219	P2.11f	240	240	229
P1.6	240	480	384	P2.13	120	120	286
P1.7	240	180	201	P2.14	120	120	288
P1.8	750	480	379	P3.1a	3000	2460	2278
P1.9	600	510	325	P3.1b	600	1020	682
P1.10	180	210	181	P3.1c	360	360	518
P1.11	180	180	182	P3.1d	120	120	363
P1.12	90	210	287	P3.2a	1500	1620	689
P1.13	150	180	311	P3.2b	1500	1620	693
P1.15	300	180	424	P4.1	600	1080	551
P1.17	180	360	181	P4.2	480	720	453
P1.18	480	1200	549	P4.3	360	420	338
P1.19	480	840	543	P4.4	600	600	563
P1.20	360	420	456	P4.5	360	240	346
P2.1	480	480	585	P4.6.1.1	360	240	570
P2.2	960	1080	995	P4.6.1.2	360	720	584
P2.3	480	480	598	P4.6.1.3	360	480	576
P2.4	360	480	289	P4.6.1.4	360	360	347
P2.5	2400	3060	2239	P4.6.1.5	240	390	408
P2.6a	60	60	237	P4.6.1.6	240	1260	396
P2.6b	120	120	170	P4.6.1.7	360	450	345
P2.6c	120	60	284	P4.6.1.9	240	300	396
P2.6d	360	60	290	P4.6.1.12	360	420	345
P2.6e	480	480	356	P4.6.2	840	720	781
P2.6f	120	120	167	P4.6.3.1	720	1200	1137
P2.6g	60	60	137	P4.6.3.2	960	840	1503
P2.6h	600	300	422	P4.6.4	600	780	942
P2.6i	180	180	205	P4.7	360	1020	345
P2.6j	60	60	139	P4.8	360	360	346
P2.7	960	840	1005	P4.9	360	360	584
P2.8a	600	600	683	P4.10	360	240	581
P2.8b	1200	1200	1224	P4.11	360	120	586
P2.8c	600	600	423	P4.12	360	240	589
P2.9	120	120	174	P4.13	1200	1200	1086
Correlation(Actual, Simulated) = 0.75992				P4.14	300	300	294

The correlation computed for the entire data for all four projects was 0.76, which is considered to be very highly correlated [19]. The actual and the simulated data were plotted on a line chart and a very similar trend was observed between the actual and simulated data as can be seen in Figure 4.1. The details of the individual senior design project's simulated results are given in Appendix D.

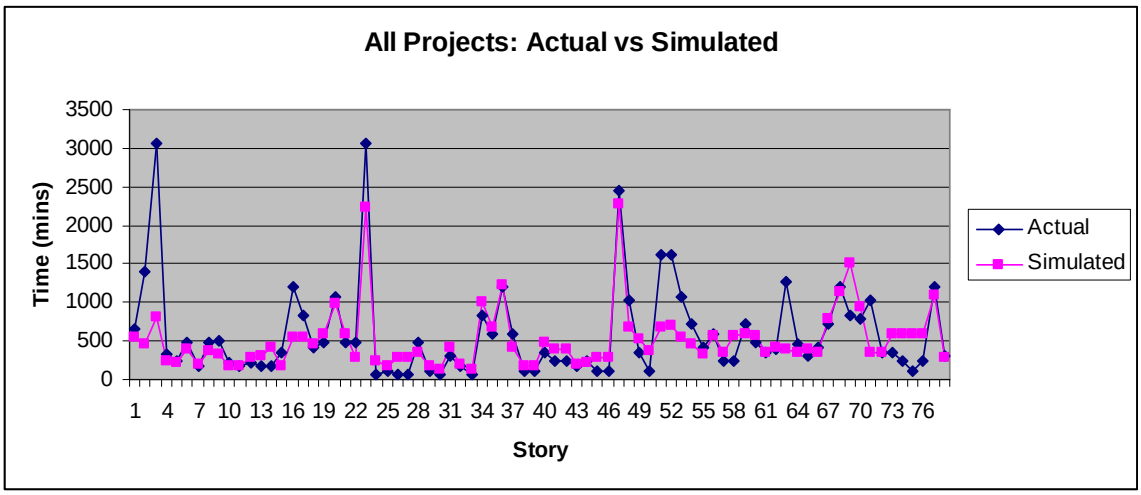

Figure 4.1: A plot showing the actual and simulated results for all projects

The regression analysis was performed using Minitab. In regression analysis, the coefficient of determination R^2 is the proportion of variability in a data set that is accounted for by a statistical model. R^2 is a statistic that will give some information about the goodness of fit of a model. In regression, the R^2 coefficient of determination is a statistical measure of how well the regression line approximates the real data points. An R^2 of 100% indicates that the regression line perfectly fits the data.

The results gave coefficient of determination $R^2 = 86.9\%$, $p < 0.0001$. This result shows a very strong statistical significance [17] and was confirmed by Dr. Maghsoodloo, who is an expert in the field of statistics. He is a retired professor in the Department of Industrial and Systems Engineering and has been very active in the field of statistics for the past two decades. The results computed are shown in Figure 4.2 and the residual plot for the results is given in Figure 4.3.

Regression Analysis

S = 90.4030 R-Sq = 86.9% R-Sq(adj) = 86.2%

Analysis of Variance

Source	**DF**	**SS**	**MS**	**F**	**P**
Regression	2	2107150	1053575	128.91	0.000
Residual Error	39	318736	8173		
Total	41	2425885			

Figure 4.2: Minitab regression analysis results

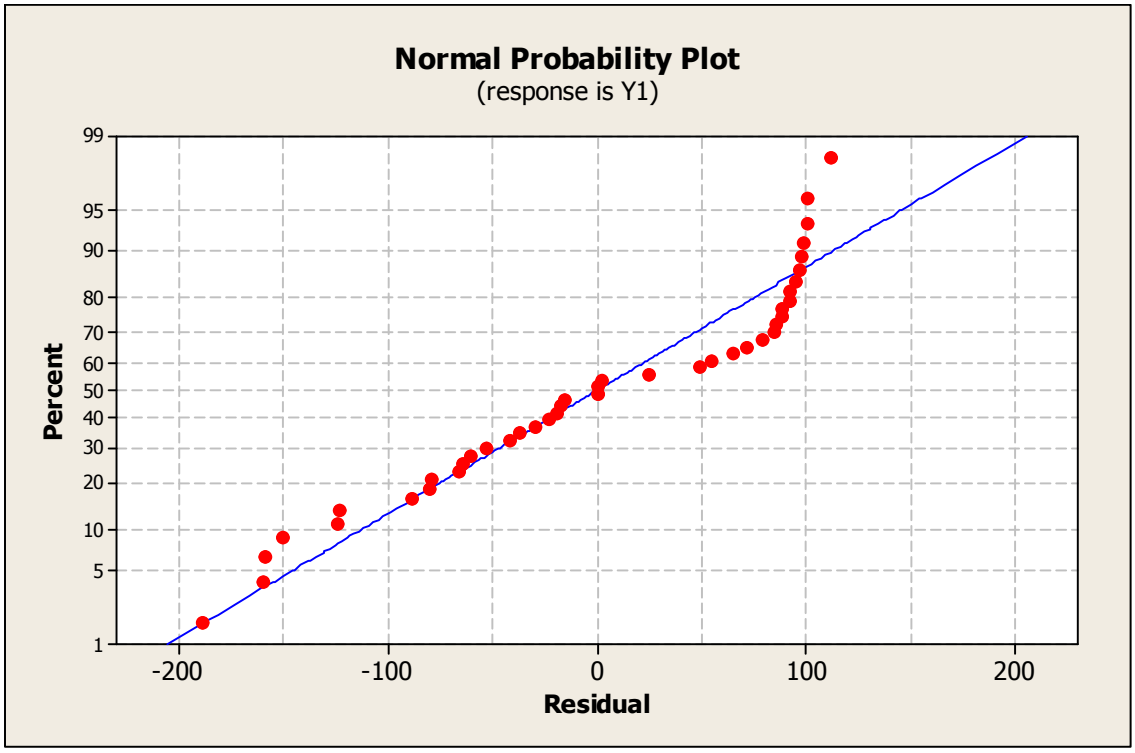

Figure 4.3: Probability plot of the residuals fall close to the straight line.

In regression analysis, least squares, also known as ordinary least squares analysis, is a method for linear regression that determines the values of unknown quantities in a statistical model by minimizing the sum of the residuals (the difference between the predicted and observed values) squared [2]. The points on the residual plot are sufficiently linear, which suggests a good fit [21] and confirms the validity of the results.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

Despite being a time-honored technique for ameliorating software development, software process simulation is not a properly-used technique. It simplifies and facilitates forecasting, training and improving customized processes. The hassle of collecting and working with bulky real-life data may not be needed; as using simulation, we can get an insight into underlying organizational trends.

5.1. Conclusions

This multi-agent, multi-threaded simulation of software development process presents a wealth of possibility to the software process community. This research showed that it is possible to simulate the performance of software development teams using performance data of individual team members from small software. The results produced by the simulation runs were favorable enough for the research to be considered useful for future work. It was demonstrated that the historical data from PSP can be successfully used to simulate and forecast the behaviors of team members in a software development team. Some other benefits of this research are:

- Simulating the development process at an individual level using agents.
- We can dynamically reconfigure our model without having to change the simulation code. This gives us the flexibility of adding or removing a

developer from the development team, enabling us to have a more accurate view of the dynamics of the corporate world.

- The interactive nature of this simulation enables the managers and developers to learn the advantages of using software process in their organization. They can use this simulation as a learning-by-doing model to learn the activities involved in software development and the associated risks with them.
- The system simulates extreme programming model of software development. XP is the most common model used in the industry today. With the prior art search done, no simulation tool was found that simulated the XP model.
- It was observed that in addition to its use as a tool to better understand and optimize the performance and/or reliability of software development, this work can also be used further to verify the applicability of development processes, and/or in teams creation.

5.2. Observations

It was observed that this research may provide several other benefits to the project leader/manager in extracting meaningful project intelligence, with:

- Standard metrics such as productivity, velocity.
- A complete history of the team's activities.
- At-a-glance indication of project status.
- Metrics that measure project status to continuously improve team processes.

Project leader can also mitigate project risk, through:

- Visibility across project status, productivity (velocity) and scope (backlog).
- A snapshot of the project in a meaningful format for the teams.
- Improved efficiency through at-a-glance project reporting that might be useful in finding the bottlenecks.
- Visibility of the team's activities.

5.2. Future Research

The fidelity of the project can be improved further by removing some of the limitations. Currently, the project only simulates one software process (eXtreme Programming), but it can be expanded to fit in several different development processes, giving the user an option to chose between different processes. The concept can be further applied to more realistic problems that have some practical utility. It will be interesting to see how this tool performs for industrial projects involving professional developers.

The results produced by the SDPTool can be verified to see if they are useful to the project leader in learning about the software process, teams' performance, and/or in making changes to the project plan. Cost factors involved in software projects can be added to this tool, allowing the estimation of the total cost of a project.

The feature of dynamically assigning tasks to team members based on their performance data and expertise by the simulation tool can be added allowing the project

leader to focus on other strategic areas. Also, suggestions to the project leader can be made by the simulation tool in formation of teams.

The resources can be better utilized if the simulation is running on parallel machines saving computational time. A transition can made from a single machine environment to multiple machines running the simulation.

REFERENCES

1. Abdel-Hamid, T., and Madnick, S., Software Project Dynamics, Prentice-Hall. Englewood Cliffs, New Jersey, 1991.

2. Abdi H., Least-squares, In M. Lewis-Beck, A. Bryman, T. Futing (Eds): Encyclopedia for research methods for the social sciences. Thousand Oaks (CA): Sage. pp. 792-795, 2003.

3. Albrecht AJ. 1979. Measuring application development productivity. Proceedings of IBM Application Development Joint SHARE/GUIDE Symposium, Monterey, CA.

4. Belscher, R., "Evaluation of Real-Time Requirements by Simulation-Based Analysis," First IEEE International Conference on Engineering of Complex Computer Systems, IEEE Computer Society Press, Los Alamitos, Calif., November 1995.

5. Birkhölzer Th., Dantas L., Dickmann C., Vaupel J., „Interactive Simulation of Software Producing Organization's Operations based on Concepts of CMMI and Balanced Scorecards," Proceedings of the 5th International Workshop on Software Process Simulation and Modelling (ProSim 2004), Edinburgh, May 2004, S. 123-132, 2004.

6. Bockle G., Clements P., McGregor J. D., Muthig D., and Schmid K., "Calculating ROI for software product lines," IEEE Software, vol. 21, pp. 23–31, May/Jun 2003.

7. Boehm B, Abts C, Brown W, Chulani S, Clark B, Horowitz E, Madachy R, Reifer D, Steece B., Software Cost Estimation with COCOMO II. Prentice Hall: NJ, USA, 2000.

8. Boehm B. W., Clark B., Horowitz E., Westland J. C., Madachy R. J., and Selby R. W., "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering, vol. 1, pp. 57–94, 1995.

9. Boehm B., Brown A. W., Madachy R., and Yang Y., "A software product line life cycle cost estimation model," in ISESE '04:The 2004 International Symposium on Empirical Software Engineering, 2004.

10. Bohem BW, Software Engineering Economics. Prentice- Hall: NJ, 1981.

11. Burke S., Radical Improvements Require Radical Actions: Simulating a High Maturity Software Organization. Technical Report, CMU/SEI-96-TR-024 ESC-TR-96-024, Carnegie Mellon University, Pittsburgh, Pennsylvania US 1997.

12. Cass AG, Lerner BS, McCall EK, Osterweil LJ, Sutton J, Stanley M, Wise AE., Little-JIL/Juliette: a process definition language and interpreter. Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, 754–757, 2000.

13. Chen Y., Gannod G. C., and Collofello J. S., A Software Product Line Process Simulator, In Proceedings of the 6th International Workshop on Software Process Simulation and Modeling (ProSim 2005), May 2005.

14. Chen Y., Gannod G. C., Collofello, and Sarjoughian H. S., "Using simulation to facilitate the study of software J. S. product line evolution," in 7th Intl. Workshop on Principles of Software Evolution, Kyoto, Japan, Sep 2004.

15. Christie A. M., Simulation: An Enabling Technology in Software Engineering, CROSSTALK – The Journal of Defense Software Engineering, pp. 2-7, April 1999.

16. Christie, A. M., "Simulation in Support of CMM-Based Process Improvement," Journal of Systems and Software.

17. Cohen J., Cohen P., West S.G., and Aiken L.S., Applied multiple regression/correlation analysis for the behavioral sciences, Hillsdale, NJ: Lawrence Erlbaum Associates, 2003.

18. Cohen S., "Predicting when product line investment pays," Software Engineering Institute, Tech. Rep. CMU/SEI-2003-TN-017, 2003.

19. Cohen, J., Statistical power analysis for the behavioral sciences, Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

20. Collofello J.S., University/Industry Collaboration in Developing a Simulation Based Software Project Management Training Course In Proceedings of the Thirteenth Conference on Software Engineering Education and Training, IEEE Computer Society: Austin, TX 161–168, 2000.

21. Cook R. D. and Weisberg S., Residuals and Influence in Regression, New York: Chapman and Hall, 1982.

22. Donzelli P. and Iazeolla G., Hybrid simulation modeling of the software process. Journal of Systems and Software, 59(3/3), 2001.

23. Drappa A. and Ludewig J. Quantitative modeling for the interaction simulation of software projects. Journal of Systems and Software, 46(2/3), 1999.

24. Drappa A. and Ludewig J., "Simulation in Software Engineering Training," Proceedings of the 22nd International Conference on Software Engineering, ICSE, Limerick, Ireland, pp. 199-208, June 2000.

25. Electronic Arts, SimCity 3000, 1998.

26. Electronic Arts, The Sims, 2000.

27. Fenton NE, Pfleeger SH, Software Metrics:A Rigorous and Practical Approach. International Thomson Computer Press: UK, 1997.

28. Ferreira S. and Collofello J., et al. Utilization of process modeling and simulation in understanding the effects of requirements volatility in software development, 2003.

29. Forrester J. W. Principles of Systems. Productivity Press, Portland, Oregon, USA, 1971.

30. Forrester, J., Industrial Dynamics, The M.I.T. Press, Cambridge, MA, 1961.

31. Frederick Brooks, "*The Mythical Man-Month"*, Addison-Wesley, 1975.

32. Gardner, L.L., M.E. Grant, and L.J. Rolston, "Using Simulation to Benchmark Traditional vs. Activity-Based Costing in Product Mix Decisions," WSC '94: Proceedings of the 1994 Conference on Winter Simulation, pp. 1050- 1057, 1994

33. Gilbert N., Computer Simulation of Social Processes. Social Research Update, Issue 6, Department of Sociology, University of Surrey, UK, 1994.

34. Hansen, G.A., "Simulating Software Development Processes," IEEE Computer, January 1996.

35. Harding A. (ed.), Microsimulation and Public Policy. Elsevier, 1996.

36. Heineman, G., Automatic Translation of Process Modeling Formalisms, Tech. Rept. CUCS-036-93, Department of Computer Science, Columbia University, New York City, New York, November 1993.

37. Host M, Regnall B, Dag J, Nedstam J, Nyberg C., Exploring bottle necks in market driven requirements management processes with discrete event simulation. Journal of Systems and Software 59(3): 323–332, 2001.

38. Howell F, McNab R., Simjava: a discrete event simulation package for Java with applications in computer systems modelling. Proceedings of the First International Conference on Web-based Modelling and Simulation. Society for Computer Simulation: San Diego, CA, 1998.

39. http://fire.nist.gov/bfrlpubs/build99/PDF/b99012.pdf, Retrieved 01/05/2006.

40. http://reliability.sandia.gov/Industrial_Engr/Stochastic_Modeling/stochastic_modeling.htm, Retrieved 01/05/2006.

41. http://sunset.usc.edu/SAMSA, Retrieved 01/05/2006.

42. http://www.decisioneering.com/monte-carlo-simulation.html, Retrieved 01/05/2006.

43. http://www.extremeprogramming.org/, Retrieved 01/05/2006.

44. http://www.prosim.pdx.edu, Retrieved 01/05/2006.

45. http://www.sei.cmu.edu/plp/modeling_costs.html, Retrieved 01/05/2006.

46. http://www.sei.cmu.edu/publications/documents/97.reports/97tr029/97tr029chap03.htm, Retrieved 01/05/2006.

47. http://www.xprogramming.com/, Retrieved 01/05/2006.

48. Humphrey W., A Discipline for Software Engineering, Addison Wesley, 1995.

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

50. Humphrey W.S. and Kellner M.I., Software Process Modeling: Principles of Entity Process Models. Proc. 11th. Intern. Conf. Software Engineering, IEEE Computer Society, Pittsburgh, PA, 331-342, 1989.

51. Joines J.A. Roberts S.D., Fundamentals of Object-Oriented Simulation, Proceedings of the 1998 Winter Simulation Conference, 1998.

52. Kahen G, Lehman M.M., Ramil J.F., and Werinck P., Dynamic Modelling in the Investigation of Policies for E-type Software Evolution. Proceedings of the Software Process Simulation Modeling Workshop (ProSim 2000), London, 2000.

53. Kellner, M., and Raffo, D., Measurement Issues in Quantitative Simulations of Process Models, in Proceedings of the Workshop on Process Modeling and Empirical Studies of Software Evolution, pp. 33-37, 1997.

54. Kellner, M., Software Process Modeling: Value and Experience, in SEI Technical Review, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, pp. 23-54, 1989.

55. Kellner, M.I., "Software Process Modeling Support for Management Planning and Control," First International Conference on the Software Process, Redondo Beach, Calif., 1991, pp. 8-28.

56. Kellner, M.I., Madachy, R.J., Raffo, D.M., "Software Process Simulation Modeling, Why? What? How?", Journal of Systems and Software, Vol. 46, No. 2-3, pp. 91-105, 1999.

57. Kocaoglu, D., Martin, R., and Raffo, D., Moving Toward a Unified Continuous and Discrete Event Simulation Model for Software Development, in Proceedings of the Silver Falls Workshop on Software Process Simulation Modeling (ProSim'98), 1998.

58. Kusumoto., S., Mizuno O., Kikuno O., Hirayama Y., Takagi Y., and Sakamoto O., A New Software Project Simulator Based on Generalized Stochastic Petri-net. Proceedings of the 19th International Conference on Software Engineering (ICSE-19), IEEE Computer Society Press, California, pp. 293-302, 1997.

59. Lakey P., A hybrid software process simulation model for project management. Proceedings of the 6th Process Simulation Modeling Workshop (ProSim 2003), Portland, Oregon, USA, 2003.

60. Lehman M. and Ramil J. F., The impact of feedback in the global software process. Journal of Systems and Software, 46(2/3), 1999.

61. Lerch, F., et al., "Using Simulation-Based Experiments for Software Requirements Engineering," N. Mead, ed., Annals of Software Engineering, Vol. 3, 1997.

62. Madachy, R., A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment, Ph.D. Dissertation, Dept. of Industrial and Systems Engineering, University of Southern California, Los Angeles, California, December 1994.

63. MAPICS Inc.. 2004, AweSim, http://www.pritsker.com/awesim.asp. Retrieved 12/03/2004.

64. Martin R. H. and Raffo D. M., Application of a hybrid process simulation model to a software development project. Journal of Systems and Software, 59(3/3), 2001.

65. Melis M., Turnu I., Cau A. and Concas G., A Software Process Simulation Model of Extreme Programming.

66. Merrill D. and Collofello J. S., Improving Software Project Management Skills Using a Software Project Simulator, Frontiers In Education Conference 1997 (FIE97), 1997.

67. Mi P. and Scacchi W., A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes. IEEE Trans. Knowledge and Data Engineering, 2(3), 283-294, 1990.

68. Mi P. and Scacchi W., Process Integration in CASE Environments. IEEE Software, 9(2), 45-53, March 1992. Reprinted in Computer-Aided Software Engineering, (2nd Edition), E. Chikofski (ed.), IEEE Computer Society, 1993.

69. Microsoft Corporation, “Microsoft project,” Jan 2005.

70. Misra J., Distributed Discrete-Event Simulation. ACM Computing Surveys, 18(1) (1986) 39-65, 1986.

71. Navarro E. Oh and Hoek A. van der, “Software Process Modeling for an Interactive, Graphical, Educational Software Engineering Simulation Game,” Proceedings of the 5th International Workshop on Software Process Simulation and Modelling (ProSim 2004), Edinburgh, May 2004, S. 171-176.

72. Navarro EO, van derHoek A., SimSE: An interactive simulation game for software engineering education. In Proceedings of the Seventh IASTED International Conference on Computers and Advanced Technology in Education, Uskov V (ed.). ACTA Press: Kauai, Hawaii, 12–17, 2004.

73. Neu H. and Hanne T., et al. Creating a code inspection model for simulation-based decision support. 2003.

74. Noll J, Scacchi W., Specifying process-oriented hypertext for organizational computing. Journal of Network and Computer Applications 24(1): 39–61, 2001.

75. Nulden U, Scheepers H., Understanding and learning about escalation: simulation in action. Proceedings of the 3rd Process Simulation ModelingWorkshop (ProSim 2000), London,UK, 2000.

76. Parunak V.D., Savit R., and Riolo R.: Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users Guide. Proceedings of Multi-Agent Systems and Agent-Based Simulation (MABS'98), Lecture Notes of Artificial Intelligence Vol. 1534, Springer Verlag, Berlin Germany, 1998.

77. Paulk, M., et al., Key Practices of the Capability Maturity Model, Version 1.1, Tech. Rept. CMU/SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, February 1993.

78. Perry, M.J., Distributed cognition and computer supported collaborative design: the organization of work in construction engineering. Unpublished PhD Thesis. Department of Information Systems and Computing, Brunel University, UK, 1998.

79. Pfahl D, Laitenberger G, Ruhe G, Dorsch J, Krivobokova T., Evaluating the learning effectiveness of using simulations in software project management education: results from a twice replicated experiment. Information and Software Technology 46: 81–147, 2004.

80. Pfahl D., Klemm M., and Ruhe G., “A CBT Module with Integrated Simulation Component for Software Project Management Education and Training,” Journal of Systems and Software 59, Elsevier, New York, 2001, pp. 283 298.

81. Putnam LH, Meyer W., Measures for Excellence: Reliable Software on Time within Budget. Prentice-Hall, NJ, 1992

82. Raffo D, Kellner M., Predicting the impact of potential process changes: a quantitative approach to process modeling. Elements of Software Process Assessment and Improvement. IEEE Computer Society Press, 1999.

83. Raffo D. M., et al. Software process simulation to achieve higher CMM levels. Journal of Systems and Software, 46(2/3), 1999.

84. Raffo D.M. and Kellner M.I.: Analysing Process Improvements Using the Process Tradeoff Analysis Method. Proceedings of the Software Process Simulation Modeling Workshop (ProSim 2000), London, 2000.

85. Raffo, D., Modeling Software Processes Quantitatively and Assessing the Impact of Potential Process Changes on Process Performance, Ph.D. Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

86. Raffo, D.M. and Kellner M.I., “Using Quantitive Process Modeling to Forecast the Impact of Potential Process Improvements,” Proceedings of the 10th International Forum on COCOMO and Software Cost Modeling, Pittsburgh, PA., October 1995.

87. Ramil J. F. and Smith N., Qualitative simulation and the study of software evolution, 2003.

88. Richardson, G., Pugh, A., Introduction to System Dynamics Modeling with DYNAMO, The M.I.T. Press, Cambridge, MA, 1981.

89. Ruiz M., Isabel R. and Toro M., An Integrated Framework for Simulation-based Software Process Improvement, Softw. Process Improve. Pract. 2004; 9: 81–93

90. Scacchi W. Experience with software process simulation and modeling. Journal of Systems and Software, 46(2/3), 1999.

91. Scacchi W., Understanding software process redesign using modeling, analysis and simulation, Softw. Process Improve. Pract. 2000; 5: 183–195, 2000.

92. Scacchi, Walt and Barry Boehm, "Virtual Systems Acquisition: Approach and Transitions," Acquisition Review Quarterly, Vol. 5, No. 2, spring 1998.

93. Schmid K. and Verlage M., "The economic impact of product line adoption and evolution," IEEE Software, vol. 19, no. 4, pp. 50 –57, Jul 2002.

94. Sharp H, Hall P. 2000. An interactive multimedia software house simulation for postgraduate software engineers. Proceedings of the 22nd International Conference on Software Engineering.ACM: Limerick, Ireland, 688–691.

95. Stallinger F. and Grunbacher P., System Dynamics Modeling and Simulation of Collaborative Requirements Engineering. Proceedings of the Software Process Simulation Modeling Workshop (ProSim 2000), London, 2000.

96. Storrle H., Making agile processes scalable. 2003.

97. Summary of CAPI-Developed Simulations for the U.S. Postal Service, http:// idt.net/~capi99/usps.htm. Retrieved 01/12/2006.

98. Tvedt, J., “An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time”, Ph.D. Dissertation, Arizona State University, Tempe, Arizona, 1996.

99. Tvedt, J.D. and J.S. Collofello, “Evaluating the Effectiveness of Process Improvements on Software Development Cycle Time via System Dynamics Modeling,” Proceedings of the 19th Annual International Computer Software and Applications Conference, 1995.

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

101. Wernick P. and Lehman M. Software process white box modelling for FEAST/1. Journal of Systems and Software, 46(2/3), 1999.

102. Wickenberg T. and Davidsson P., On Multi Agent Based Simulation of Software Development Processes, Third International Workshop on Multi-Agent Based Simulation, MABS'02, 15 July 2002

103. Zeigler B. P. and Sarjoughian H. S., “Introduction to DEVS modeling & simulation with Java,” Aug 2003.

104. Zeigler B.P., Object Oriented Simulation with Hierarchical Modular Models, Academic Press, 1990.

APPENDICES

APPENDIX A - SDPTOOL INSTRUCTION SUMMARY

SDPTool: Software Development Process Tool, Version 1.0

Authors: Ravikant Agarwal, Dr. David Umphress (Committee Chair).

Auburn University, AL, USA

Process Simulated: A version of eXtreme Programming used by the students of COMP 4710 (Senior Design Project).

Inputs: PSP historical data of each team member/employee (total of 4 members in team developing a software product working in teams of 2 each.)

Description:

For demo purposes, there is a file 'agent.txt' that contains PSP data for 4 employees/team members, which you can use to run this program. In case you use this file, you will have to choose an option during the initialization mode of this product that allows you to use an existing agent file. You can replace the data on this file by any PSP data that you want to use. This product also allows you to create a new file with member's PSP data. In that case, you can select the appropriate radio button that allows you to create a new file.

In case of creating a new file, you will be presented with a form that allows you to enter the PSP data needed for each individual member in the development team. You can also specify the number of team members (for COMP 4710, it will be 4), which should be an even number so that pair programming can be done. You will need to initialize the number of user stories, project size (if you know the LOC count, else leave it as Not

Applicable "NA"), and the time interval you would like the simulation to pause for you to check on the ongoing development process.

Upon initialization, the simulation starts and expects the user to enter the estimated values for each user story as and when it's being developed during the run-time. The simulation also enables the user to select the measurement matrices that he/she wants the simulation to report at the end of each time interval. The user would be working as a project leader and would be creating teams by selecting team members and assigning deferent user stories to each team.

The simulation would present the user with the project progress statistics at the end of each iteration cycle. The user can select individual employee, team, or user story to view its stats. At each time interval, the user can update the project by changing the user story, development priority list, adding, deleting, or changing the estimated size/duration of any user story, or by modifying the teams.

The simulation would stop when all the user stories are completed or when the user wishes to end the simulation. The output files generated as a result of the simulation would be saved in SDPTool folder.

APPENDIX B – AGENT'S PSP DATA*

Project 1:

1, Agent1, 504, 23, 10.53, 23, 8, 0.8358, 23.8, 5.9, 0, 33.9, 1.3, 9.4, 18.2, 7.5, 0, 0, 23, 0, 0, 0, 0, 0, 0, 10, 13, 0, 6.1739, 6.1739, 6.1739, 3.5, 8.23, 6.1739
2, Agent2, 595, 69, 26.73, 72, 8, 0.8113, 21.9, 2, 0, 34.6, 0.6, 7.9, 27.6, 5.4, 0, 0, 72, 0, 0, 0, 0, 0, 0, 49, 20, 0, 5.986, 5.986, 5.986, 1.918, 16.5, 5.986
3, Agent3, 310, 19, 15.5388, 19, 8, 1.0823, 13.2, 10.5, 0, 52.1, 0, 7.8, 11.7, 4.7, 0, 3, 12, 1, 2, 1, 0, 0, 5, 10, 3, 1, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6
4, Agent4, 441, 88, 9.72, 88, 8, 2.435, 20.9, 16.7, 0.2, 22.2, 0.7, 5.1, 25.2, 9, 0, 0, 68, 11, 9, 0, 0, 0, 3, 59, 26, 0, 3.7, 3.7, 3.7, 2.1, 7.8, 3.7

Project 2:

1, Agent1, 621, 21, 15.17, 32.49, 22, 1.099, 1.069, 32.8, 11, 1, 20.2, 1, 1.4, 23.1, 9.6, 0, 0, 21, 0, 1, 0, 0, 0, 4, 5, 12, 0, 10.136, 10.136, 10.136, 2.6, 16.67, 10.136
2, Agent2, 436, 17, 9.23, 16.91, 17, 0, 1.091, 14.3, 16.7, 1.8, 29.6, 2.7, 14.4, 13.4, 7.2, 0, 1, 2, 7, 7, 0, 0, 1, 3, 6, 7, 0, 4.43, 4.43, 4.43, 3.14, 7, 4.43
3, Agent3, 384, 20, 18.51, 37.14, 20, 50.22, 0.17, 25.4, 10.7, 0, 33.3, 0.7, 4.4, 21.5, 4, 0, 0, 0, 7, 13, 0, 0, 0, 1, 7, 12, 0, 3.45, 3.45, 3.45, 1.28, 4.62, 3.45
4, Agent4, 886, 37, 6.41, 25.6, 51, 118.95, 0.082, 5.6, 9, 0.2, 37.7, 2.7, 22.1, 11.7, 11, 0, 3, 42, 3, 3, 0, 0, 0, 0, 22, 15, 0, 6.82, 6.82, 6.82, 3, 17.4667, 6.82

Project 3:

1, Agent1, 673, 27, 10.44, 22.43, 27, 155.25, -0.26, 25.3, 7.8, 0, 25, 0, 10.6, 21.1, 10.3, 0, 0, 27, 0, 0, 0, 0, 0, 0, 25, 2, 0, 4.28, 4.28, 4.28, 4.28, 4.28, 4.28
2, Agent2, 880, 29, 8.18, 26.21, 29, 47.92, 0.74, 12.5, 17.6, 1.4, 32.2, 1.8, 1.4, 12.8, 20.3, 2, 5, 19, 1, 1, 1, 0, 4, 16, 3, 5, 1, 19.68, 19.68, 19.68, 15, 22, 19.68
3, Agent3, 563, 35, 21, 51.96, 38, 34, 0.39, 18.1, 12.7, 2.1, 30.3, 0.6, 3.1, 13.3, 19.8, 7, 8, 22, 0, 0, 1, 1, 3, 9, 18, 4, 0, 1.23, 1.23, 1.23, 1, 1.25, 1.23
4, Agent4, 628, 19, 16.97, 33.6, 23, 43.06, 0.81, 21.1, 5.9, 0.1, 26.3, 1.6, 3.6, 33.7, 7.7, 0, 3, 20, 0, 0, 0, 0, 0, 0, 6, 13, 0, 0, 0, 0, 0, 0, 0

Project 4:

1, Agent1, 578, 58, 17.4, 29.26, 58, 0, 1.024, 31.1, 7.3, 0.8, 24.1, 0.7, 2.6, 20.9, 12, 0, 4, 45, 1, 8, 0, 0, 4, 20, 13, 21, 0, 3.12, 3.12, 3.12, 1.38, 6.09, 3.12
2, Agent2, 725, 62, 8.96, 19.8, 62, 0, 1, 12, 7.1, 0, 25.5, 2.2, 2.6, 35.1, 15.6, 0, 0, 62, 0, 0, 0, 0, 0, 12, 17, 33, 0, 8.95, 8.95, 8.95, 2.23, 28.82, 8.95
3, Agent3, 896, 91, 12.49, 12.49, 105, 0, 1, 21.8, 6.3, 0.4, 29.1, 2.5, 4.7, 28.9, 6.3, 1, 3, 89, 1, 11, 0, 0, 1, 9, 35, 46, 0, 9.67, 9.67, 9.67, 4.91, 16.30, 9.67
4, Agent4, 242, 29, 9, 11.82, 29, 22.44, 0.51, 31.5, 17, 1.2, 17.2, 0, 3.8, 16.3, 12.3, 0, 6, 18, 2, 3, 0, 0, 0, 3, 16, 10, 0, 9.83, 9.83, 9.83, 2.44, 25.3, 9.83

* The student names have been omitted for student's confidentiality

APPENDIX C – STORY ASSIGNMENTS FOR TEAMS

Project 1:

Teams	Story Name
Team1	4, 5, 7, 8, 9, 10, 11, 15
Team2	1, 2, 3, 6, 12, 13, 17, 18, 19, 20

Project 2:

Teams	Story Name
Team1	1, 2, 3, 5, 6a, 6c, 7, 8a, 8b, 11a, 11b, 11c, 13, 14
Team2	4, 6b, 6d, 6e, 6f, 6g, 6h, 6i, 6j, 8c, 9, 10, 11d, 11f

Project 3:

Teams	Story Name
Team1	1a, 1b, 1c, 1d
Team2	2a, 2b

Project 4:

Teams	Story Name
Team1	1, 2, 3, 4, 5, 6.1.4, 6.1.7, 6.1.9, 6.1.12, 6.2, 7, 8, 13, 14
Team2	6.1.1, 6.1.2, 6.1.3, 6.1.5, 6.1.6, 6.3.1, 6.3.2, 6.4, 9, 10, 11, 12

APPENDIX D – SIMULATION RESULTS

Project 1:

Table C.1: Comparison of Planned, Actual and Simulation Results for Project 1

	Time (mins)		
Story Name	**Planned**	**Actual**	**Simulated**
1	480	660	549
2	360	1410	454
3	900	3060	805
4	360	330	239
5	300	240	219
6	240	480	384
7	240	180	201
8	750	480	379
9	600	510	325
10	180	210	181
11	180	180	182
12	90	210	287
13	150	180	311
15	300	180	424
17	180	360	181
18	480	1200	549
19	480	840	543
20	360	420	456

Correlation(Actual, Simulated) = 0.82459

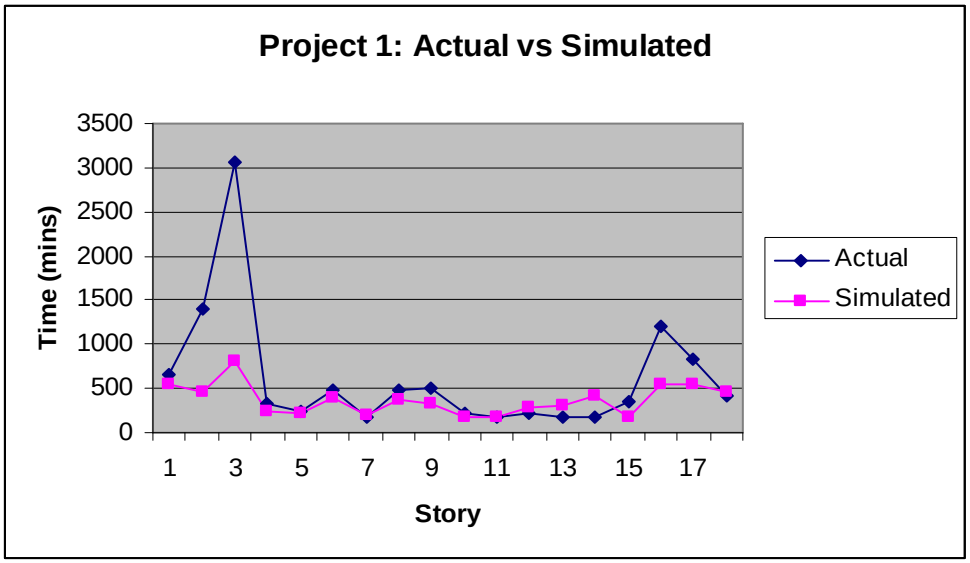

Figure C.1: A plot showing the actual and simulated results for Project 1

Project 2:

Table C.2: Comparison of Planned, Actual and Simulation Results for Project 2

	Time (mins)		
Story Name	**Planned**	**Actual**	**Simulated**
1	480	480	585
2	960	1080	995
3	480	480	598
4	360	480	289
5	2400	3060	2239
6a	60	60	237
6b	120	120	170
6c	120	60	284
6d	360	60	290
6e	480	480	356
6f	120	120	167
6g	60	60	137
6h	600	300	422
6i	180	180	205
6j	60	60	139
7	960	840	1005
8a	600	600	683
8b	1200	1200	1224
8c	600	600	423
9	120	120	174
10	120	120	172
11a	360	360	486
11b	240	240	393
11c	240	240	390
11d	180	180	201
11f	240	240	229
13	120	120	286
14	120	120	288

Correlation(Actual, Simulated) = 0.969119

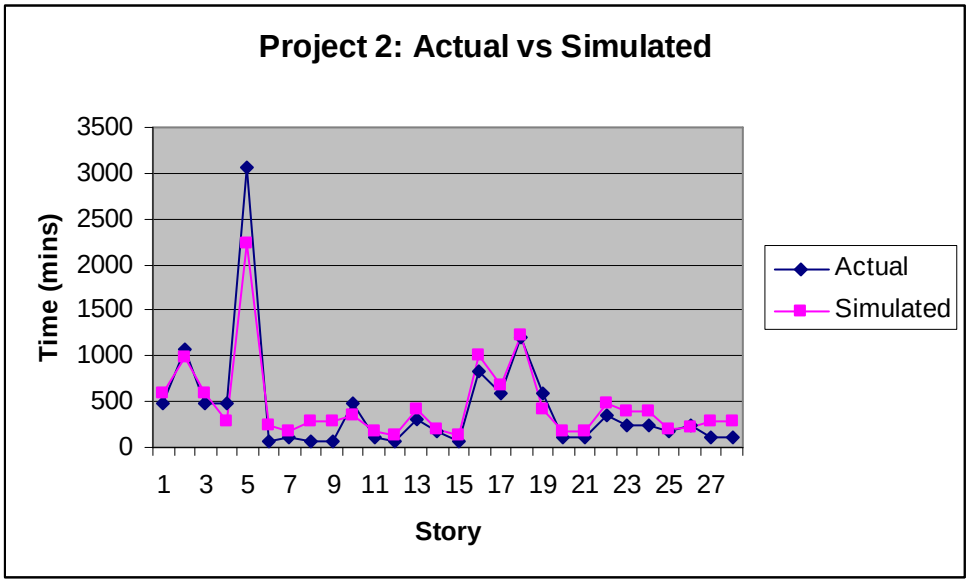

Figure C.2: A plot showing the actual and simulated results for Project 2

Project 3:

Table C.3: Comparison of Planned, Actual and Simulation Results for Project 3

	Time (mins)		
Story Name	**Planned**	**Actual**	**Simulated**
1a	3000	2460	2278
1b	600	1020	682
1c	360	360	518
1d	120	120	363
2a	1500	1620	689
2b	1500	1620	693

Correlation(Actual, Simulated) = 0.812687

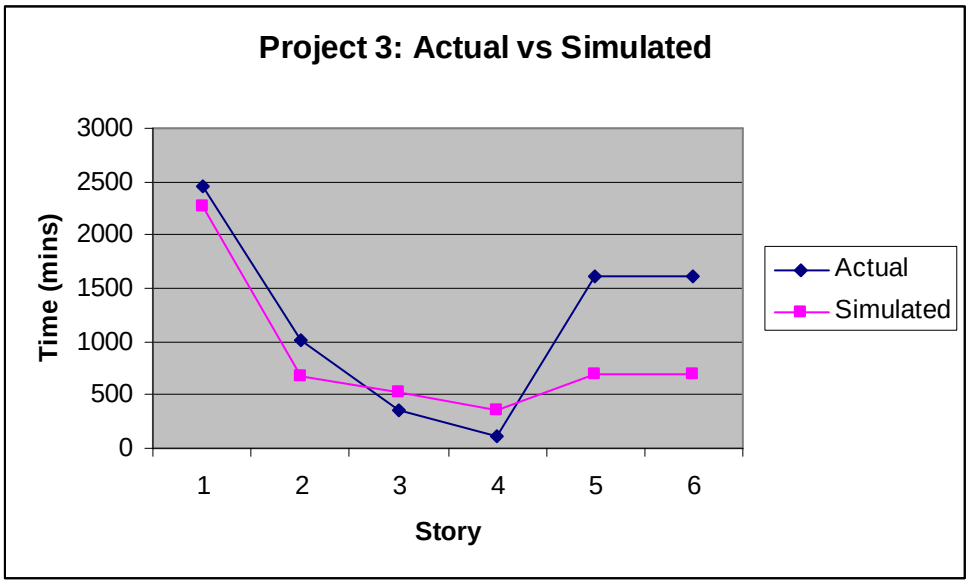

Figure C.3: A plot showing the actual and simulated results for Project 3

Project 4:

Table C.4: Comparison of Planned, Actual and Simulation Results for Project 4

	Time (mins)		
Story Name	**Planned**	**Actual**	**Simulated**
1	600	1080	551
2	480	720	453
3	360	420	338
4	600	600	563
5	360	240	346
6.1.1	360	240	570
6.1.2	360	720	584
6.1.3	360	480	576
6.1.4	360	360	347
6.1.5	240	390	408
6.1.6	240	1260	396
6.1.7	360	450	345
6.1.9	240	300	396
6.1.12	360	420	345
6.2	840	720	781
6.3.1	720	1200	1137
6.3.2	960	840	1503
6.4	600	780	942
7	360	1020	345
8	360	360	346
9	360	360	584
10	360	240	581
11	360	120	586
12	360	240	589
13	1200	1200	1086
14	300	300	294

Correlation(Actual, Simulated) = 0.495018

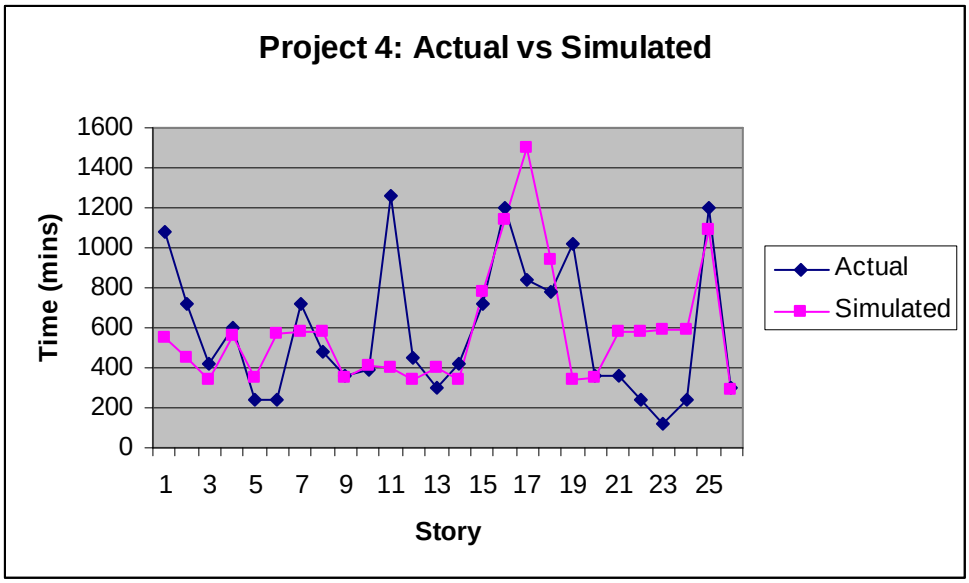

Figure C.4: A plot showing the actual and simulated results for Project 4

APPENDIX E – SCREENSHOTS AND DESCRIPTIONS

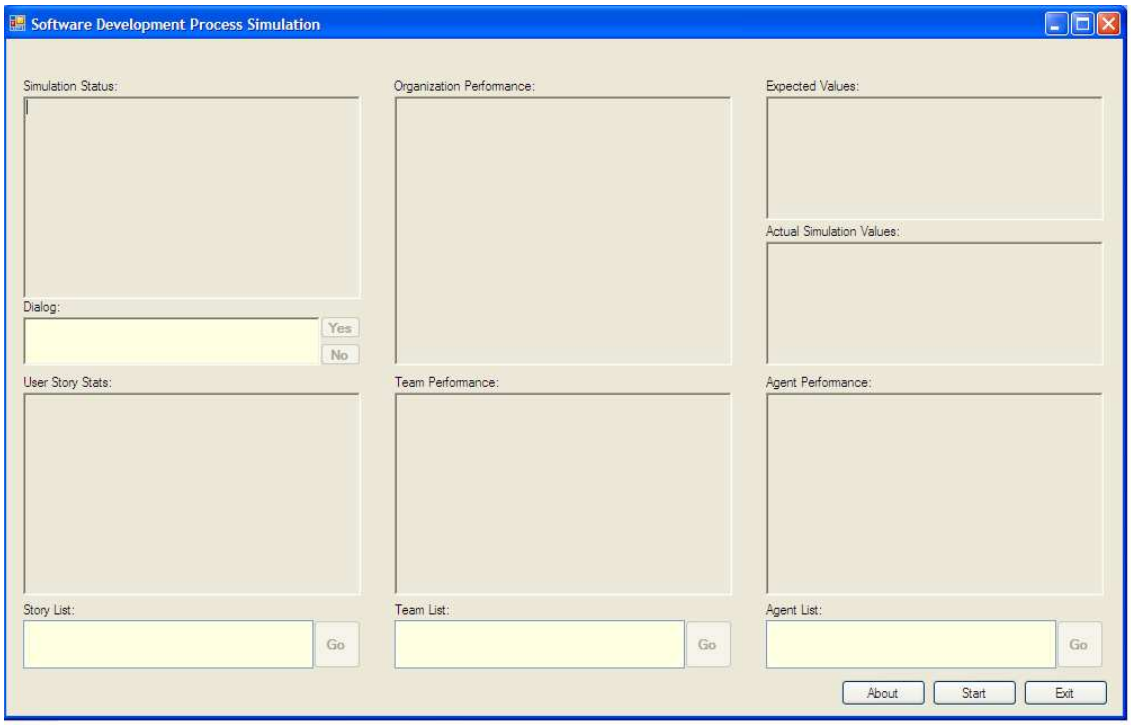

Figure1: SDPTool: The Main Window

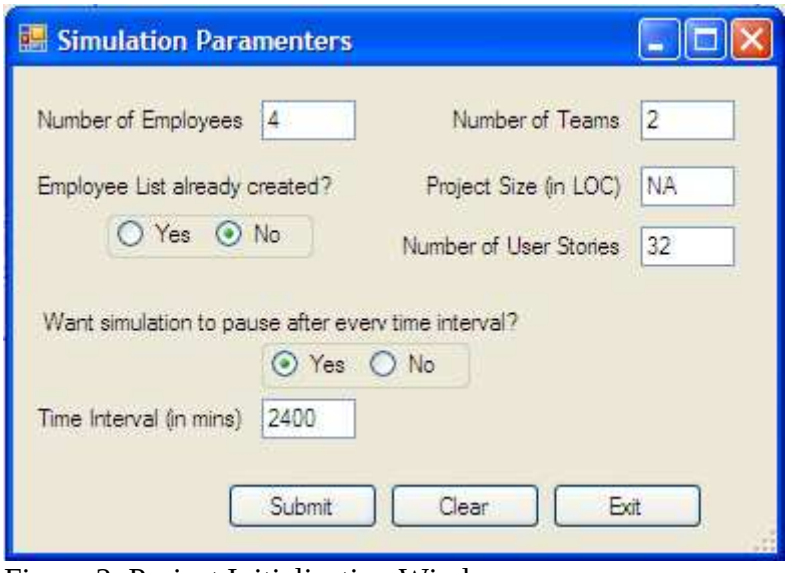

Figure 2: Project Initialization Window

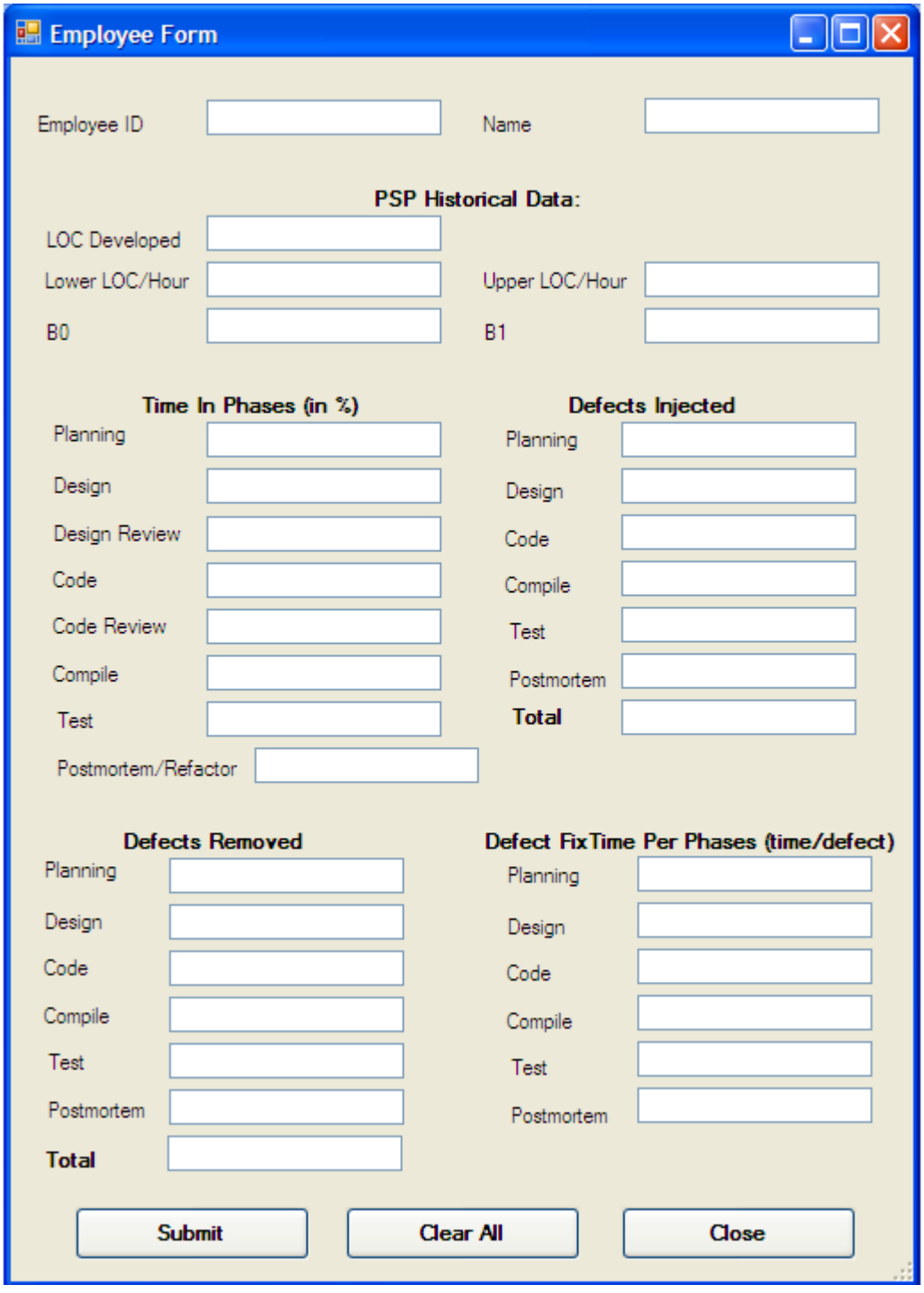

Figure 3: Agent's PSP Data Entry Form

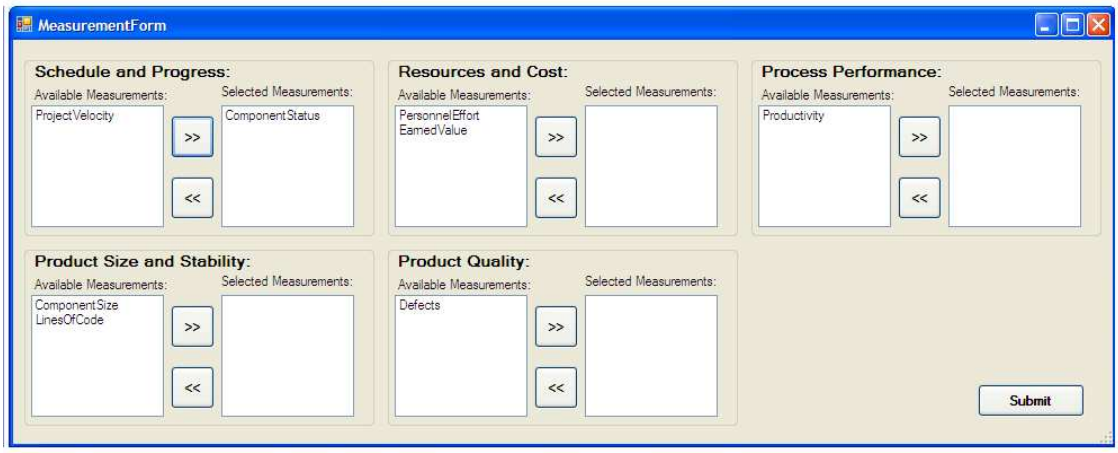

Figure 4: Measurement form

Figure 5: Form for Entering the Stories

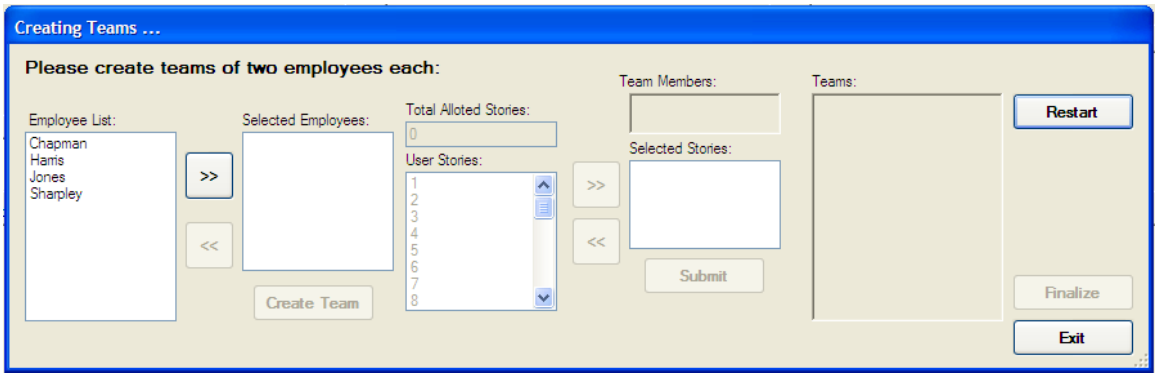

Figure 6: Team Creation form

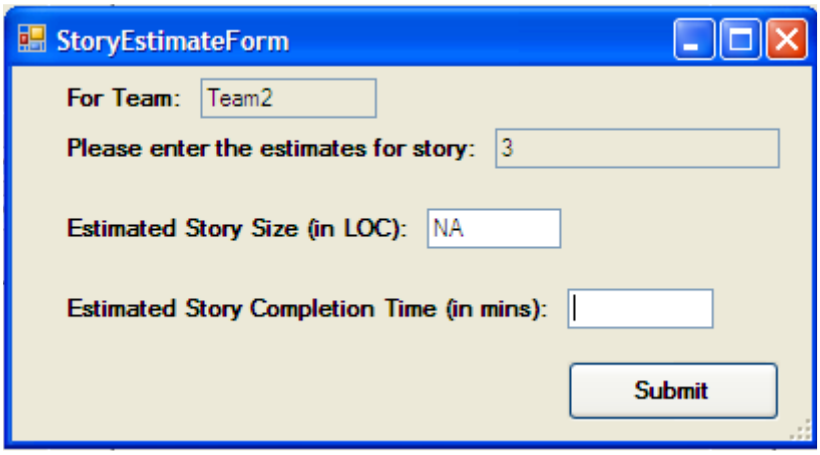

Figure 7: Story Estimation Form

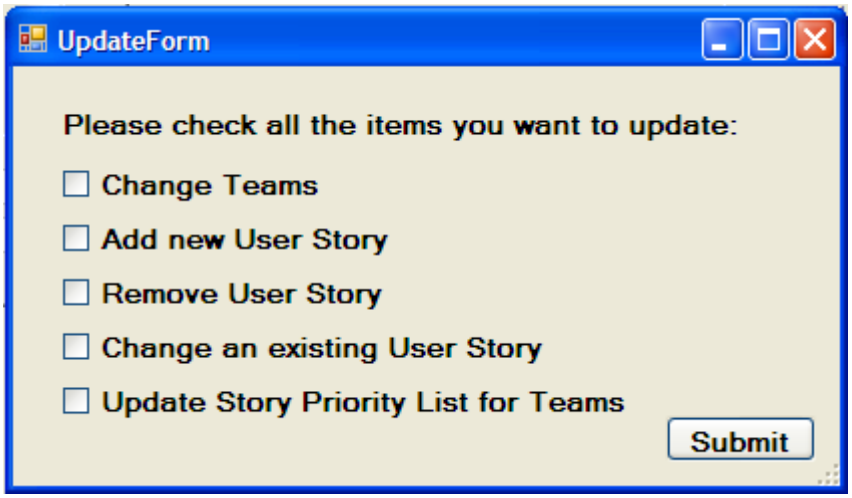

Figure 8: Project/Development Update Form

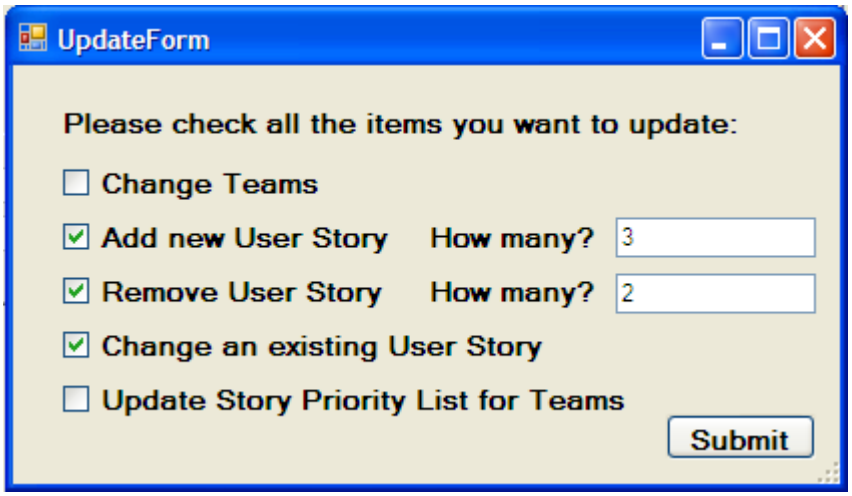

Figure 9: Project/Development Update Form showing that a user can define the number of stories to be modified

Figure 10: The story selection form

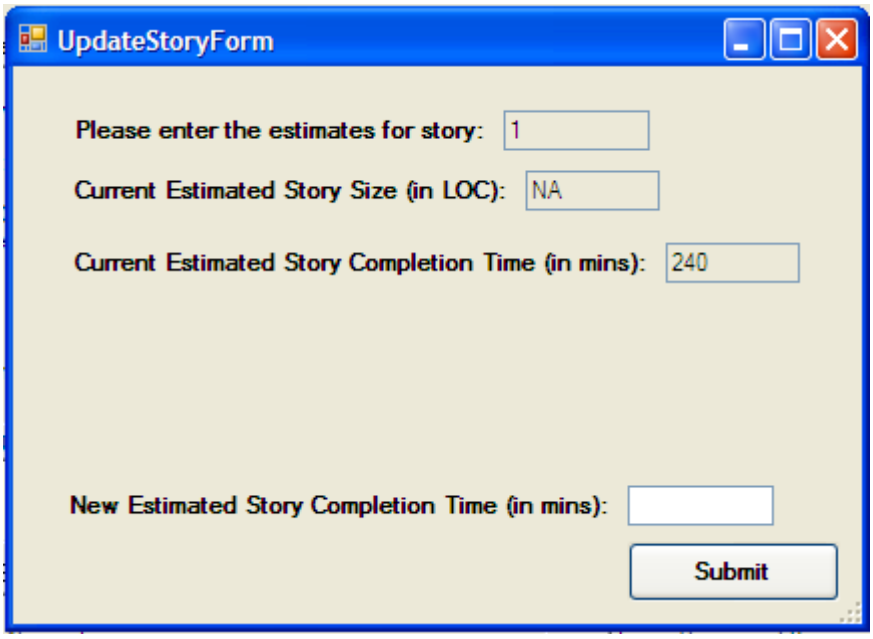

Figure 11: Story update form

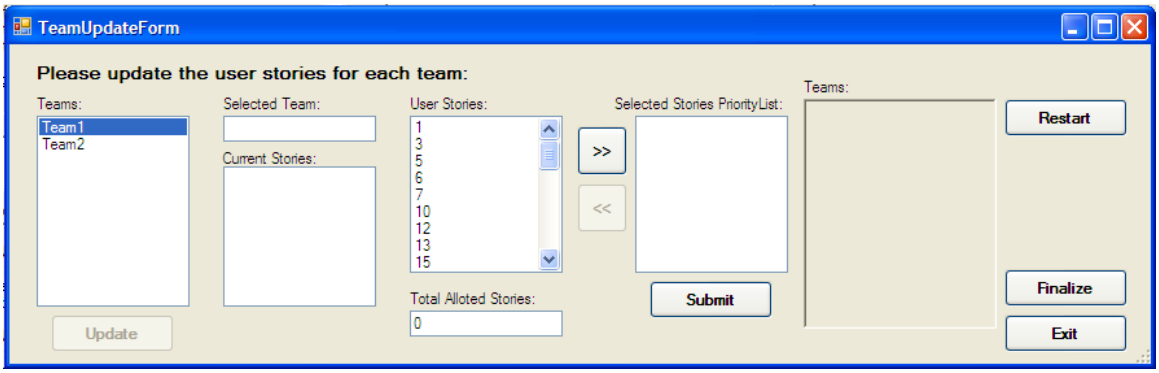

Figure 12: Team update form

Figure 13: Story addition form

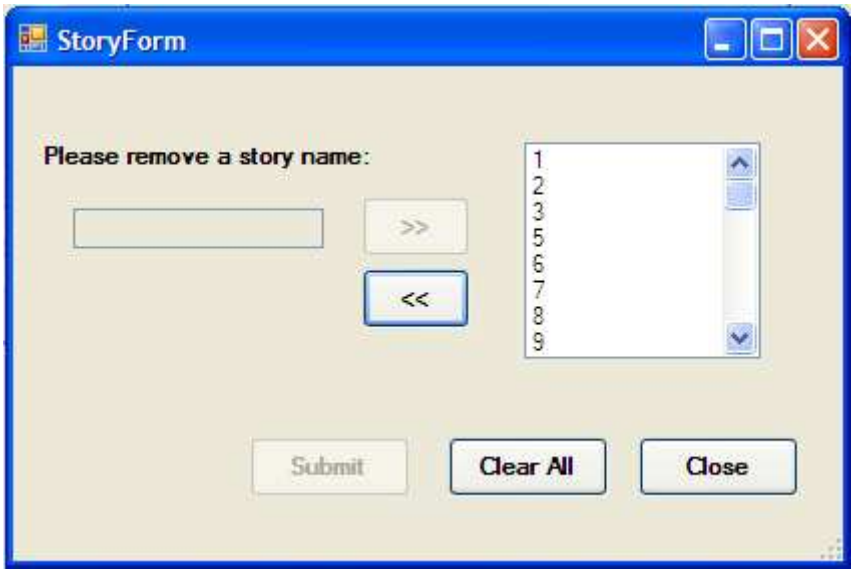

Figure 14: Story removal form

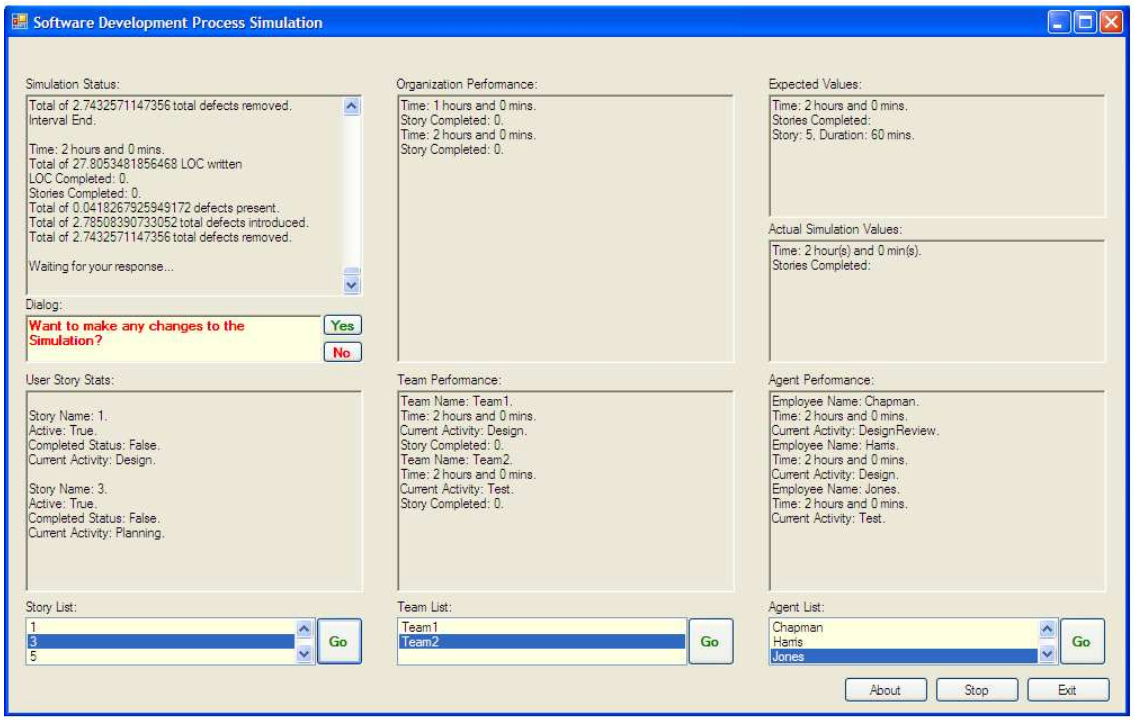

Figure 15: SDPTool showing the current status of the project

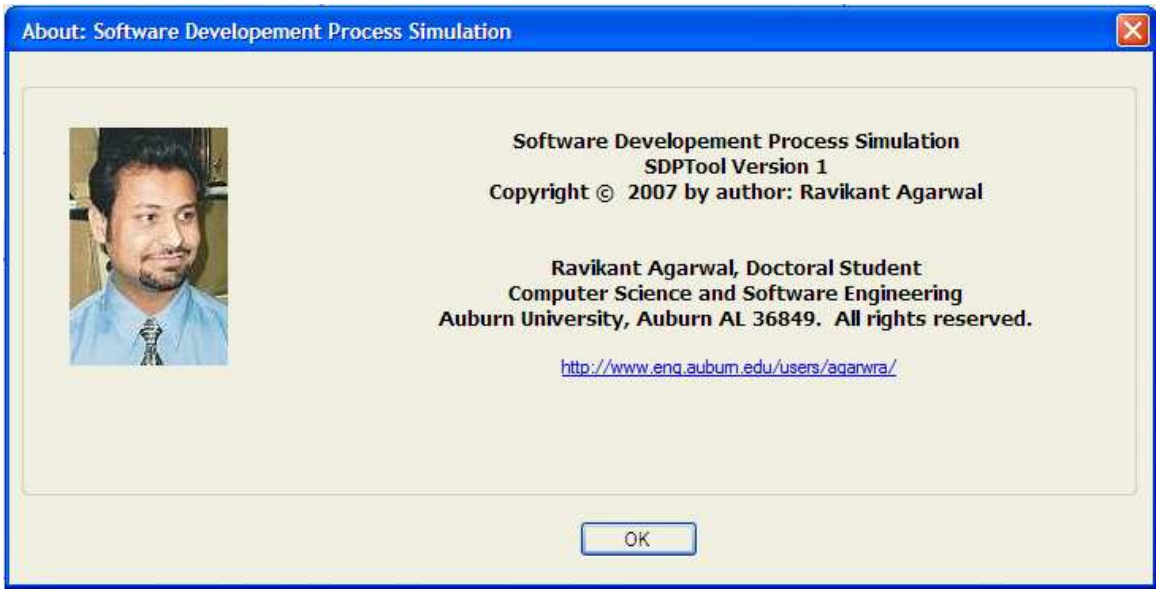

Figure 16: The About window

APPENDIX F – SOURCE CODE

```
using System;
using System.IO;
using System.Threading;
using System.Windows.Forms;

namespace SDPTool
{
   class UserStory
   {
      int id;
      string name;
      double estimatedSize;
      double estimatedDuration;
      double size;
      double b0 = 0;
      double b1 = 1;
      string activityType;
      Team parent;
      bool complete;
      bool active;
      bool locKnown;
      double locPlanned = 0;
      double locDesigned = 0;
      double locDesignReviewed = 0;
      double locWritten = 0;
      double locCodeReviewed = 0;
      double locCompiled = 0;
      double locTested = 0;
      double locRefactored = 0;
      double locCompleted = 0;
      double currentDefects = 0;
      double timeSpent = 0;

      public int Id
      {
         get  //get accessor method
         {
            return id;
         }
         set   //set accessor method
         {
            id = value;
         }
      }

      public string Name
      {
         get  //get accessor method
         {
            return name;
```

```
    }
    set   //set accessor method
    {
        name = value;
    }
}

public string ActivityType
{
    get  //get accessor method
    {
        return activityType;
    }
    set   //set accessor method
    {
        activityType = value;
    }
}

public Team Parent
{
    get  //get accessor method
    {
        return parent;
    }
    set   //set accessor method
    {
        parent = value;
    }
}

public double Size
{
    get  //get accessor method
    {
        return size;
    }
    set   //set accessor method
    {
        size = value;
    }
}

public bool LocKnown
{
    get  //get accessor method
    {
        return locKnown;
    }
    set   //set accessor method
    {
        locKnown = value;
    }
```

```
}

public double B0
{
   get  //get accessor method
   {
      return b0;
   }
   set   //set accessor method
   {
      b0 = value;
   }
}

public double B1
{
   get  //get accessor method
   {
      return b1;
   }
   set   //set accessor method
   {
      b1 = value;
   }
}

public double EstimatedSize
{
   get  //get accessor method
   {
      return estimatedSize;
   }
   set   //set accessor method
   {
      estimatedSize = value;
   }
}

public double EstimatedDuration
{
   get  //get accessor method
   {
      return estimatedDuration;
   }
   set   //set accessor method
   {
      estimatedDuration = value;
   }
}

public double TimeSpent
{
   get  //get accessor method
```

```
    {
        return timeSpent;
    }
    set   //set accessor method
    {
        timeSpent = value;
    }
}

public double LocPlanned
{
    get  //get accessor method
    {
        return locPlanned;
    }
    set   //set accessor method
    {
        locPlanned = value;
    }
}

public double LocDesigned
{
    get  //get accessor method
    {
        return locDesigned;
    }
    set   //set accessor method
    {
        locDesigned = value;
    }
}

public double LocDesignReviewed
{
    get  //get accessor method
    {
        return locDesignReviewed;
    }
    set   //set accessor method
    {
        locDesignReviewed = value;
    }
}

public double LocWritten
{
    get  //get accessor method
    {
        return locWritten;
    }
    set   //set accessor method
    {
```

```
        locWritten = value;
    }
}

public double LocCodeReviewed
{
    get  //get accessor method
    {
        return locCodeReviewed;
    }
    set   //set accessor method
    {
        locCodeReviewed = value;
    }
}

public double LocCompiled
{
    get  //get accessor method
    {
        return locCompiled;
    }
    set   //set accessor method
    {
        locCompiled = value;
    }
}

public double LocTested
{
    get  //get accessor method
    {
        return locTested;
    }
    set   //set accessor method
    {
        locTested = value;
    }
}

public double LocRefactored
{
    get  //get accessor method
    {
        return locRefactored;
    }
    set   //set accessor method
    {
        locRefactored = value;
    }
}

public double LocCompleted
```

```
{
   get  //get accessor method
   {
      return locCompleted;
   }
   set   //set accessor method
   {
      locCompleted = value;
   }
}

public double CurrentDefects
{
   get  //get accessor method
   {
      return currentDefects;
   }
   set   //set accessor method
   {
      currentDefects = value;
   }
}

public bool Complete
{
   get  //get accessor method
   {
      return complete;
   }
   set   //set accessor method
   {
      complete = value;
   }
}

public bool Active
{
   get  //get accessor method
   {
      return active;
   }
   set   //set accessor method
   {
      active = value;
   }
}

public UserStory(int ID, string Name, double eSize, double duration)
{
   id = ID;
   name = Name;
   estimatedSize = eSize;
   estimatedDuration = duration;
```

```
            size = b0 + EstimatedSize * b1;
            activityType = "Planning";
            complete = false;
            active = true;
        }

        public void UpdateSize(double newLoc, double locDeleted)
        {
            double newEstimatedSize = estimatedSize - locDeleted;

            if (newEstimatedSize < 0)
                newEstimatedSize = 0;

            if (locPlanned > 0)
            {
                locPlanned -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locPlanned;
            }

            if (locDesigned > 0)
            {
                locDesigned -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locDesigned;
            }

            if (locDesignReviewed > 0)
            {
                locDesignReviewed -= ((estimatedSize - newEstimatedSize) / estimatedSize) *
locDesignReviewed;
            }

            if (locWritten > 0)
            {
                locWritten -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locWritten;
            }

            if (locCodeReviewed > 0)
            {
                locCodeReviewed -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locCodeReviewed;
            }

            if (locCompiled > 0)
            {
                locCompiled -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locCompiled;
            }

            if (locTested > 0)
            {
                locTested -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locTested;
            }

            if (locRefactored > 0)
            {
                locRefactored -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locRefactored;
            }
```

```
    if (locCompleted > 0)
    {
        locCompleted -= ((estimatedSize - newEstimatedSize) / estimatedSize) * locCompleted;
    }

    estimatedSize = newEstimatedSize + newLoc;
    size = b0 + EstimatedSize * b1;
    if (locCompleted < size)
    {
        complete = false;
    }
    else
    {
        locCompleted = size;
        complete = true;
        activityType = "Completed";
    }
}

public void UpdateActivity()
{
    //Check pre-requisites for the assigned activity and update if needed
    switch (activityType)
    {
        case "Planning":
            if (locPlanned >= size)
            {
                activityType = "Design";
            }
            break;
        case "Design":
            if ((locDesigned >= size) && (locDesignReviewed >= size))
            {
                activityType = "Code";
            }
            break;
        case "Code":
            if ((locWritten >= size) && (locCodeReviewed >= size))
            {
                activityType = "Compile";
            }
            break;
        case "Compile":
            if (locCompiled >= size)
            {
                activityType = "Test";
            }
            break;
        case "Test":
            if (locTested >= size)
            {
                if (currentDefects > 0)
```

```
                activityType = "Debug";
            else
                activityType = "Refactor";
        }
        break;
    case "Debug":
        if (currentDefects <= 0)
        {
            currentDefects = 0;
            activityType = "Refactor";
        }
        break;
    case "Refactor":
        if (currentDefects > 0)
            activityType = "Debug";
        else if (locRefactored >= size)
        {
            complete = true;
            activityType = "Completed";
        }
        break;
  }
  if (currentDefects < 0)
    currentDefects = 0;
}


public void Update()
{
  size = b0 + estimatedSize * b1;
  activityType = "Planning";
  if (locPlanned > size)
  {
    locPlanned = size;
  }

  if (locDesigned > size)
  {
    locDesigned = size;
  }

  if (locDesignReviewed > size)
  {
    locDesignReviewed = size;
  }
  if (locWritten > size)
  {
    locWritten = size;
  }

  if (locCodeReviewed > size)
  {
    locCodeReviewed = size;
```

```
            }
            if (locCompiled > size)
            {
               locCompiled = size;
            }

            if (locTested > size)
            {
               locTested = size;
            }

            if (locRefactored > size)
            {
               locRefactored = size;
               locCompleted = size;
               complete = true;
            }

            if (locCompleted < size)
            {
               complete = false;
            }
            else
            {
               locCompleted = size;
               complete = true;
               activityType = "Completed";
            }
         }
      }
   }
```

```
namespace SDPTool
{
   partial class UpdateTeamPriorityForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.label5 = new System.Windows.Forms.Label();
         this.teamTextBox = new System.Windows.Forms.TextBox();
         this.UpdateButton = new System.Windows.Forms.Button();
         this.label2 = new System.Windows.Forms.Label();
         this.storyListBox = new System.Windows.Forms.ListBox();
         this.teamListBox = new System.Windows.Forms.ListBox();
         this.label1 = new System.Windows.Forms.Label();
         this.removeStoryButton = new System.Windows.Forms.Button();
         this.label8 = new System.Windows.Forms.Label();
         this.selectedStoryListBox = new System.Windows.Forms.ListBox();
         this.selectStoryButton = new System.Windows.Forms.Button();
         this.restartButton = new System.Windows.Forms.Button();
         this.exitButton = new System.Windows.Forms.Button();
         this.submitButton = new System.Windows.Forms.Button();
         this.finalizeButton = new System.Windows.Forms.Button();
         this.label4 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.teamSummaryBox = new System.Windows.Forms.RichTextBox();
         this.SuspendLayout();
         //
         // label5
```

```
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(163, 39);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(82, 13);
            this.label5.TabIndex = 80;
            this.label5.Text = "Selected Team:";
            //
            // teamTextBox
            //
            this.teamTextBox.Location = new System.Drawing.Point(166, 55);
            this.teamTextBox.Name = "teamTextBox";
            this.teamTextBox.Size = new System.Drawing.Size(118, 20);
            this.teamTextBox.TabIndex = 81;
            //
            // UpdateButton
            //
            this.UpdateButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.UpdateButton.Location = new System.Drawing.Point(31, 208);
            this.UpdateButton.Name = "UpdateButton";
            this.UpdateButton.Size = new System.Drawing.Size(95, 29);
            this.UpdateButton.TabIndex = 79;
            this.UpdateButton.Text = "Update";
            this.UpdateButton.UseVisualStyleBackColor = true;
            this.UpdateButton.Click += new System.EventHandler(this.UpdateButton_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(312, 39);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(79, 13);
            this.label2.TabIndex = 77;
            this.label2.Text = "Current Stories:";
            //
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(315, 55);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 121);
            this.storyListBox.TabIndex = 78;
            //
            // teamListBox
            //
            this.teamListBox.FormattingEnabled = true;
            this.teamListBox.Location = new System.Drawing.Point(22, 55);
            this.teamListBox.Name = "teamListBox";
            this.teamListBox.Size = new System.Drawing.Size(118, 147);
            this.teamListBox.TabIndex = 76;
            //
            // label1
```

```
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(19, 39);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(42, 13);
            this.label1.TabIndex = 75;
            this.label1.Text = "Teams:";
            //
            // removeStoryButton
            //
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(439, 107);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 72;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(481, 39);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(112, 13);
            this.label8.TabIndex = 73;
            this.label8.Text = "User Story Priority List:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(484, 55);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 121);
            this.selectedStoryListBox.TabIndex = 74;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(439, 63);
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 71;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
            // restartButton
            //
            this.restartButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.restartButton.Location = new System.Drawing.Point(769, 42);
            this.restartButton.Name = "restartButton";
            this.restartButton.Size = new System.Drawing.Size(95, 29);
            this.restartButton.TabIndex = 68;
            this.restartButton.Text = "Restart";
            this.restartButton.UseVisualStyleBackColor = true;
            this.restartButton.Click += new System.EventHandler(this.restartButton_Click);
            //
            // exitButton
            //
            this.exitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.exitButton.Location = new System.Drawing.Point(769, 208);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(95, 29);
            this.exitButton.TabIndex = 67;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(498, 182);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 63;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // finalizeButton
            //
            this.finalizeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.finalizeButton.Location = new System.Drawing.Point(769, 173);
            this.finalizeButton.Name = "finalizeButton";
            this.finalizeButton.Size = new System.Drawing.Size(95, 29);
            this.finalizeButton.TabIndex = 66;
            this.finalizeButton.Text = "Finalize";
            this.finalizeButton.UseVisualStyleBackColor = true;
            this.finalizeButton.Click += new System.EventHandler(this.finalizeButton_Click);
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(19, 13);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(341, 17);
            this.label4.TabIndex = 60;
```

```
    this.label4.Text = "Please update the user stories for each team:";
    //
    // label3
    //
    this.label3.AutoSize = true;
    this.label3.Location = new System.Drawing.Point(611, 26);
    this.label3.Name = "label3";
    this.label3.Size = new System.Drawing.Size(42, 13);
    this.label3.TabIndex = 64;
    this.label3.Text = "Teams:";
    //
    // teamSummaryBox
    //
    this.teamSummaryBox.Location = new System.Drawing.Point(614, 42);
    this.teamSummaryBox.Name = "teamSummaryBox";
    this.teamSummaryBox.ReadOnly = true;
    this.teamSummaryBox.Size = new System.Drawing.Size(149, 177);
    this.teamSummaryBox.TabIndex = 65;
    this.teamSummaryBox.Text = "";
    //
    // UpdateTeamPriorityForm
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(890, 245);
    this.Controls.Add(this.label5);
    this.Controls.Add(this.teamTextBox);
    this.Controls.Add(this.UpdateButton);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.storyListBox);
    this.Controls.Add(this.teamListBox);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.removeStoryButton);
    this.Controls.Add(this.label8);
    this.Controls.Add(this.selectedStoryListBox);
    this.Controls.Add(this.selectStoryButton);
    this.Controls.Add(this.restartButton);
    this.Controls.Add(this.exitButton);
    this.Controls.Add(this.submitButton);
    this.Controls.Add(this.finalizeButton);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.teamSummaryBox);
    this.Name = "UpdateTeamPriorityForm";
    this.Text = "UpdateTeamPriorityForm";
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label5;
```

```
        private System.Windows.Forms.TextBox teamTextBox;
        private System.Windows.Forms.Button UpdateButton;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox storyListBox;
        private System.Windows.Forms.ListBox teamListBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button removeStoryButton;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.ListBox selectedStoryListBox;
        private System.Windows.Forms.Button selectStoryButton;
        private System.Windows.Forms.Button restartButton;
        private System.Windows.Forms.Button exitButton;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Button finalizeButton;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.RichTextBox teamSummaryBox;
    }
}
```

```
namespace SDPTool
{
   partial class UpdateStoryForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.label3 = new System.Windows.Forms.Label();
         this.textBox4 = new System.Windows.Forms.TextBox();
         this.label2 = new System.Windows.Forms.Label();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.submitButton = new System.Windows.Forms.Button();
         this.label1 = new System.Windows.Forms.Label();
         this.textBox3 = new System.Windows.Forms.TextBox();
         this.label5 = new System.Windows.Forms.Label();
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.label4 = new System.Windows.Forms.Label();
         this.textBox5 = new System.Windows.Forms.TextBox();
         this.label6 = new System.Windows.Forms.Label();
         this.textBox6 = new System.Windows.Forms.TextBox();
         this.label7 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // label3
         //
         this.label3.AutoSize = true;
         this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.label3.Location = new System.Drawing.Point(28, 91);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(292, 13);
            this.label3.TabIndex = 4;
            this.label3.Text = "Current Estimated Story Completion Time (in mins):";
            //
            // textBox4
            //
            this.textBox4.Location = new System.Drawing.Point(151, 145);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(73, 20);
            this.textBox4.TabIndex = 7;
            this.textBox4.Text = "NA";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.Location = new System.Drawing.Point(28, 55);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(222, 13);
            this.label2.TabIndex = 2;
            this.label2.Text = "Current Estimated Story Size (in LOC):";
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(256, 52);
            this.textBox2.Name = "textBox2";
            this.textBox2.ReadOnly = true;
            this.textBox2.Size = new System.Drawing.Size(67, 20);
            this.textBox2.TabIndex = 3;
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(307, 237);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(106, 30);
            this.submitButton.TabIndex = 12;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(28, 25);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(211, 13);
```

```
            this.label1.TabIndex = 0;
            this.label1.Text = "Please enter the estimates for story:";
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(326, 88);
            this.textBox3.Name = "textBox3";
            this.textBox3.ReadOnly = true;
            this.textBox3.Size = new System.Drawing.Size(67, 20);
            this.textBox3.TabIndex = 5;
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label5.Location = new System.Drawing.Point(29, 181);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(137, 13);
            this.label5.TabIndex = 8;
            this.label5.Text = "New LOC to be Added:";
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(245, 22);
            this.textBox1.Name = "textBox1";
            this.textBox1.ReadOnly = true;
            this.textBox1.Size = new System.Drawing.Size(73, 20);
            this.textBox1.TabIndex = 1;
            this.textBox1.Text = "UserStory";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(29, 148);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(116, 13);
            this.label4.TabIndex = 6;
            this.label4.Text = "LOC to be Deleted:";
            //
            // textBox5
            //
            this.textBox5.Location = new System.Drawing.Point(172, 178);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(67, 20);
            this.textBox5.TabIndex = 9;
            this.textBox5.Text = "NA";
            //
            // label6
            //
            this.label6.AutoSize = true;
```

```
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label6.Location = new System.Drawing.Point(25, 212);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(276, 13);
            this.label6.TabIndex = 10;
            this.label6.Text = "New Estimated Story Completion Time (in mins):";
            //
            // textBox6
            //
            this.textBox6.Location = new System.Drawing.Point(307, 209);
            this.textBox6.Name = "textBox6";
            this.textBox6.Size = new System.Drawing.Size(73, 20);
            this.textBox6.TabIndex = 11;
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(26, 120);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(222, 13);
            this.label7.TabIndex = 13;
            this.label7.Text = "Enter LOC if known, else leave blank:";
            //
            // UpdateStoryForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(424, 278);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.textBox6);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.textBox5);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.textBox3);
            this.Name = "UpdateStoryForm";
            this.Text = "UpdateStoryForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion
```

```
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox textBox6;
        private System.Windows.Forms.Label label7;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class UpdateTeamPriorityForm : Form
   {

      static int MAX_STORY = 50;
      FileStream teamFile = null;
      StreamWriter sw = null;
      string folderPath;
      string fileName;
      int totalTeams = 0;
      int totalStory = 0;
      string[] storyName = new string[MAX_STORY];
      int storyCount = -1;


      public string FolderPath
      {
         get  //get accessor method
         {
            return folderPath;
         }
         set   //set accessor method
         {
            folderPath = value;
         }
      }

      public UpdateTeamPriorityForm(int tCount, int sCount)
      {
         InitializeComponent();
         UpdateButton.Enabled = true;
         storyListBox.Enabled = false;
         submitButton.Enabled = false;
         selectStoryButton.Enabled = true;
         removeStoryButton.Enabled = false;
         finalizeButton.Enabled = true;
         fileName = folderPath + "updateteams.txt";
         try
         {
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
         }
```

```
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
                MessageBox.Show("File team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            totalTeams = tCount;
            totalStory = sCount;
        }

        public int TotalTeams
        {
            get
            {
                return totalTeams;
            }
            set
            {
                totalTeams = value;
            }
        }

        public void AddTeam(string team)
        {
            teamListBox.Items.Add(team);
        }

        public void AddStory(string story)
        {
            storyListBox.Items.Add(story);
            storyName[++storyCount] = story;
        }

        private void restartButton_Click(object sender, EventArgs e)
        {

            storyListBox.Items.Clear();
            selectedStoryListBox.Items.Clear();

            teamSummaryBox.Text = null;

            submitButton.Enabled = false;

            try
            {
                if (teamFile != null)
                    teamFile.Close();
                teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
                sw = new StreamWriter(teamFile);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
```

```
            MessageBox.Show("File: team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }


        for (int i = 0; i <= storyCount; i++)
        {
            storyListBox.Items.Add(storyName[i]);
        }
    }

    private void finalizeButton_Click(object sender, EventArgs e)
    {
        if (teamFile != null)
        {
            sw.Close();
            teamFile.Close();
        }
        Close();
    }

    private void submitButton_Click(object sender, EventArgs e)
    {
        teamSummaryBox.AppendText(string.Concat("Team: ", teamTextBox.Text.Trim(), ":\n"));
        sw.Write(teamTextBox.Text.Trim());

        int sCount = 0;
        foreach (object item in selectedStoryListBox.Items)
        {
            teamSummaryBox.AppendText(string.Concat("UserStory  ", Convert.ToString(++sCount), ": ",
(string)item, "\n"));
            sw.Write(string.Concat(", ", (string)item));
        }


        selectedStoryListBox.Items.Clear();

        teamTextBox.Text = null;
        submitButton.Enabled = false;
        storyListBox.Enabled = false;
        selectedStoryListBox.Enabled = false;
        selectStoryButton.Enabled = false;
        removeStoryButton.Enabled = false;
        UpdateButton.Enabled = true;
        teamSummaryBox.ScrollToCaret();
    }


    private void exitButton_Click(object sender, EventArgs e)
    {
        this.Close();
        Environment.Exit(0);
    }
```

```
        private void selectStoryButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
                selectedStoryListBox.Items.Add(selectedText);
                storyListBox.Items.Remove(storyListBox.SelectedItem);
                removeStoryButton.Enabled = true;
            }
            else
            {
                MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void removeStoryButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedStoryListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedStoryListBox.Items.Remove(selectedItem);
                storyListBox.Items.Add(selectedItem);

                if (selectedStoryListBox.Items.Count == 0)
                    removeStoryButton.Enabled = false;
            }
        }

        private void UpdateButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)teamListBox.SelectedItem;
            if (selectedItem != null)
            {
                storyListBox.Items.Clear();
                storyListBox.Enabled = true;
                submitButton.Enabled = true;
                selectStoryButton.Enabled = true;
                removeStoryButton.Enabled = false;
                teamTextBox.Text = selectedItem;
                UpdateButton.Enabled = false;
            }
        }
    }
}
```

```
namespace SDPTool
{
   partial class UpdateTeamForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.removeStoryButton = new System.Windows.Forms.Button();
         this.label8 = new System.Windows.Forms.Label();
         this.label9 = new System.Windows.Forms.Label();
         this.selectedStoryListBox = new System.Windows.Forms.ListBox();
         this.selectStoryButton = new System.Windows.Forms.Button();
         this.storyListBox = new System.Windows.Forms.ListBox();
         this.restartButton = new System.Windows.Forms.Button();
         this.exitButton = new System.Windows.Forms.Button();
         this.label7 = new System.Windows.Forms.Label();
         this.storyCountTextBox = new System.Windows.Forms.TextBox();
         this.submitButton = new System.Windows.Forms.Button();
         this.finalizeButton = new System.Windows.Forms.Button();
         this.label4 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.teamSummaryBox = new System.Windows.Forms.RichTextBox();
         this.label1 = new System.Windows.Forms.Label();
         this.teamListBox = new System.Windows.Forms.ListBox();
         this.label2 = new System.Windows.Forms.Label();
         this.currentStorylistBox = new System.Windows.Forms.ListBox();
         this.UpdateButton = new System.Windows.Forms.Button();
         this.label5 = new System.Windows.Forms.Label();
```

```
            this.teamTextBox = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // removeStoryButton
            //
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(438, 107);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 49;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(464, 39);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(137, 13);
            this.label8.TabIndex = 50;
            this.label8.Text = "Selected Stories PriorityList:";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(306, 39);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(67, 13);
            this.label9.TabIndex = 46;
            this.label9.Text = "User Stories:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(483, 55);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 121);
            this.selectedStoryListBox.TabIndex = 51;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(438, 63);
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 48;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
```

```
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(309, 55);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 121);
            this.storyListBox.TabIndex = 47;
            //
            // restartButton
            //
            this.restartButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.restartButton.Location = new System.Drawing.Point(768, 42);
            this.restartButton.Name = "restartButton";
            this.restartButton.Size = new System.Drawing.Size(95, 29);
            this.restartButton.TabIndex = 45;
            this.restartButton.Text = "Restart";
            this.restartButton.UseVisualStyleBackColor = true;
            this.restartButton.Click += new System.EventHandler(this.restartButton_Click);
            //
            // exitButton
            //
            this.exitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.exitButton.Location = new System.Drawing.Point(768, 208);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(95, 29);
            this.exitButton.TabIndex = 44;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(306, 189);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(104, 13);
            this.label7.TabIndex = 38;
            this.label7.Text = "Total Alloted Stories:";
            //
            // storyCountTextBox
            //
            this.storyCountTextBox.Location = new System.Drawing.Point(309, 205);
            this.storyCountTextBox.Name = "storyCountTextBox";
            this.storyCountTextBox.Size = new System.Drawing.Size(118, 20);
            this.storyCountTextBox.TabIndex = 39;
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(494, 182);
```

```
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 40;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // finalizeButton
            //
            this.finalizeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.finalizeButton.Location = new System.Drawing.Point(768, 173);
            this.finalizeButton.Name = "finalizeButton";
            this.finalizeButton.Size = new System.Drawing.Size(95, 29);
            this.finalizeButton.TabIndex = 43;
            this.finalizeButton.Text = "Finalize";
            this.finalizeButton.UseVisualStyleBackColor = true;
            this.finalizeButton.Click += new System.EventHandler(this.finalizeButton_Click);
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(18, 13);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(341, 17);
            this.label4.TabIndex = 26;
            this.label4.Text = "Please update the user stories for each team:";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(610, 26);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(42, 13);
            this.label3.TabIndex = 41;
            this.label3.Text = "Teams:";
            //
            // teamSummaryBox
            //
            this.teamSummaryBox.Location = new System.Drawing.Point(613, 42);
            this.teamSummaryBox.Name = "teamSummaryBox";
            this.teamSummaryBox.ReadOnly = true;
            this.teamSummaryBox.Size = new System.Drawing.Size(149, 177);
            this.teamSummaryBox.TabIndex = 42;
            this.teamSummaryBox.Text = "";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(18, 39);
            this.label1.Name = "label1";
```

```
            this.label1.Size = new System.Drawing.Size(42, 13);
            this.label1.TabIndex = 53;
            this.label1.Text = "Teams:";
            //
            // teamListBox
            //
            this.teamListBox.FormattingEnabled = true;
            this.teamListBox.Location = new System.Drawing.Point(21, 55);
            this.teamListBox.Name = "teamListBox";
            this.teamListBox.Size = new System.Drawing.Size(118, 147);
            this.teamListBox.TabIndex = 54;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(162, 81);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(79, 13);
            this.label2.TabIndex = 55;
            this.label2.Text = "Current Stories:";
            //
            // currentStorylistBox
            //
            this.currentStorylistBox.FormattingEnabled = true;
            this.currentStorylistBox.Location = new System.Drawing.Point(165, 94);
            this.currentStorylistBox.Name = "currentStorylistBox";
            this.currentStorylistBox.Size = new System.Drawing.Size(118, 108);
            this.currentStorylistBox.TabIndex = 56;
            //
            // UpdateButton
            //
            this.UpdateButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.UpdateButton.Location = new System.Drawing.Point(32, 208);
            this.UpdateButton.Name = "UpdateButton";
            this.UpdateButton.Size = new System.Drawing.Size(95, 29);
            this.UpdateButton.TabIndex = 57;
            this.UpdateButton.Text = "Update";
            this.UpdateButton.UseVisualStyleBackColor = true;
            this.UpdateButton.Click += new System.EventHandler(this.UpdateButton_Click);
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(162, 39);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(82, 13);
            this.label5.TabIndex = 58;
            this.label5.Text = "Selected Team:";
            //
            // teamTextBox
            //
            this.teamTextBox.Location = new System.Drawing.Point(165, 55);
```

```
    this.teamTextBox.Name = "teamTextBox";
    this.teamTextBox.Size = new System.Drawing.Size(118, 20);
    this.teamTextBox.TabIndex = 59;
    //
    // UpdateTeamForm
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(881, 246);
    this.Controls.Add(this.label5);
    this.Controls.Add(this.teamTextBox);
    this.Controls.Add(this.UpdateButton);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.currentStorylistBox);
    this.Controls.Add(this.teamListBox);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.removeStoryButton);
    this.Controls.Add(this.label8);
    this.Controls.Add(this.label9);
    this.Controls.Add(this.selectedStoryListBox);
    this.Controls.Add(this.selectStoryButton);
    this.Controls.Add(this.storyListBox);
    this.Controls.Add(this.restartButton);
    this.Controls.Add(this.exitButton);
    this.Controls.Add(this.label7);
    this.Controls.Add(this.storyCountTextBox);
    this.Controls.Add(this.submitButton);
    this.Controls.Add(this.finalizeButton);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.teamSummaryBox);
    this.Name = "UpdateTeamForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
    this.Text = "TeamUpdateForm";
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button removeStoryButton;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.ListBox selectedStoryListBox;
private System.Windows.Forms.Button selectStoryButton;
private System.Windows.Forms.ListBox storyListBox;
private System.Windows.Forms.Button restartButton;
private System.Windows.Forms.Button exitButton;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.TextBox storyCountTextBox;
private System.Windows.Forms.Button submitButton;
private System.Windows.Forms.Button finalizeButton;
```

```
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.RichTextBox teamSummaryBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ListBox teamListBox;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox currentStorylistBox;
        private System.Windows.Forms.Button UpdateButton;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox teamTextBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class UpdateTeamForm : Form
   {
      static int MAX_STORY = 50;
      FileStream teamFile = null;
      StreamWriter sw = null;
      string folderPath;
      string fileName;
      int totalTeams = 0;
      int totalStory = 0;
      string[] storyName = new string[MAX_STORY];
      int storyCount = -1;

      public string FolderPath
      {
         get  //get accessor method
         {
            return folderPath;
         }
         set   //set accessor method
         {
            folderPath = value;
         }
      }
      public UpdateTeamForm(int tCount, int sCount)
      {
         InitializeComponent();
         UpdateButton.Enabled = true;
         storyCountTextBox.Text = "0";
         storyCountTextBox.Enabled = false;
         storyListBox.Enabled = false;
         submitButton.Enabled = false;
         selectStoryButton.Enabled = true;
         removeStoryButton.Enabled = false;
         finalizeButton.Enabled = true;
         fileName = folderPath + "updateteams.txt";
         try
         {
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
         }
         catch (Exception e)
```

```
            {
                Console.WriteLine(e.ToString());
                MessageBox.Show("File team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            totalTeams = tCount;
            totalStory = sCount;
        }

        public int TotalTeams
        {
            get
            {
                return totalTeams;
            }
            set
            {
                totalTeams = value;
            }
        }

        public void AddTeam(string team)
        {
            teamListBox.Items.Add(team);
        }

        public void AddStory(string story)
        {
            storyListBox.Items.Add(story);
            storyName[++storyCount] = story;
        }

        private void restartButton_Click(object sender, EventArgs e)
        {

            storyListBox.Items.Clear();
            selectedStoryListBox.Items.Clear();

            teamSummaryBox.Text = null;

            storyCountTextBox.Text = "0";
            storyCountTextBox.Enabled = false;
            submitButton.Enabled = false;

            try
            {
                if (teamFile != null)
                    teamFile.Close();
                teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
                sw = new StreamWriter(teamFile);
            }
            catch (Exception ex)
            {
```

```
            Console.WriteLine(ex.ToString());
            MessageBox.Show("File: team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }


        for (int i = 0; i <= storyCount; i++)
        {
            storyListBox.Items.Add(storyName[i]);
        }
    }

    private void finalizeButton_Click(object sender, EventArgs e)
    {
        if (teamFile != null)
        {
            sw.Close();
            teamFile.Close();
        }
        Close();
    }

    private void submitButton_Click(object sender, EventArgs e)
    {
        teamSummaryBox.AppendText(string.Concat("Team: ", teamTextBox.Text.Trim(), ":\n"));
        sw.Write(teamTextBox.Text.Trim());

        int sCount = 0;
        foreach (object item in selectedStoryListBox.Items)
        {
            teamSummaryBox.AppendText(string.Concat("UserStory  ", Convert.ToString(++sCount), ": ",
(string)item, "\n"));
            sw.Write(string.Concat(", ", (string)item));
        }

        teamSummaryBox.AppendText(string.Concat("Alloted # of Stories: ", storyCountTextBox.Text,
"\n\n"));

        sw.Write(string.Concat(", ", storyCountTextBox.Text));
        sw.Write("\n");

        selectedStoryListBox.Items.Clear();

        storyCountTextBox.Text = "0";
        storyCountTextBox.Enabled = false;
        teamTextBox.Text = null;
        submitButton.Enabled = false;
        storyListBox.Enabled = false;
        selectedStoryListBox.Enabled = false;
        selectStoryButton.Enabled = false;
        removeStoryButton.Enabled = false;
        UpdateButton.Enabled = true;
        teamSummaryBox.ScrollToCaret();
```

```
        }


        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
            Environment.Exit(0);
        }

        private void selectStoryButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
                selectedStoryListBox.Items.Add(selectedText);
                storyListBox.Items.Remove(storyListBox.SelectedItem);
                removeStoryButton.Enabled = true;
                storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) + 1);
            }
            else
            {
                MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void removeStoryButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedStoryListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedStoryListBox.Items.Remove(selectedItem);
                storyListBox.Items.Add(selectedItem);
                storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) - 1);

                if (selectedStoryListBox.Items.Count == 0)
                    removeStoryButton.Enabled = false;
            }
        }

        private void UpdateButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)teamListBox.SelectedItem;
            if (selectedItem != null)
            {
                currentStorylistBox.Items.Clear();
                storyCountTextBox.Enabled = true;
                storyListBox.Enabled = true;
                submitButton.Enabled = true;
                selectStoryButton.Enabled = true;
                removeStoryButton.Enabled = false;
                teamTextBox.Text = selectedItem;
                UpdateButton.Enabled = false;
```

```
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SDPTool
{
   public partial class UpdateStoryForm : Form
   {
      double locAdded;
      double locDeleted;
      double estimatedDuration;
      double locPerHour;
      double estimatedSize;
      bool locKnown;

      public double LocAdded
      {
         get  //get accessor method
         {
            return locAdded;
         }
      }

      public double LocDeleted
      {
         get  //get accessor method
         {
            return locDeleted;
         }
      }

      public double EstimatedDuration
      {
         get  //get accessor method
         {
            return estimatedDuration;
         }
      }

      public UpdateStoryForm(string storyName, double size, double duration, bool lKnown, double
estimatedLocPerHour)
      {
         InitializeComponent();
         textBox1.Text = storyName;
         estimatedSize = size;
         locKnown = lKnown;
         locPerHour = estimatedLocPerHour;
         if (locKnown.Equals(true))
```

```
            {
                textBox2.Text = Convert.ToString(size);
            }
            else
            {
                textBox2.Text = "NA";
                label4.Visible = false;
                label5.Visible = false;
                label7.Visible = false;
                textBox4.Visible = false;
                textBox5.Visible = false;
            }
            textBox3.Text = Convert.ToString(duration);
        }

        public UpdateStoryForm(string storyName)
        {
            InitializeComponent();
            textBox1.Text = storyName;
        }

        private void submitButton_Click(object sender, EventArgs e)
        {
            estimatedDuration = Convert.ToDouble(textBox6.Text);
            //LOC was not used and just duration was known
            if (locKnown.Equals(false))
            {
                if (Convert.ToDouble(textBox3.Text.Trim()) > estimatedDuration)
                {
                    locDeleted = (Convert.ToDouble(textBox3.Text.Trim()) - estimatedDuration) * locPerHour;
                    locAdded = 0;
                }
                else if (Convert.ToDouble(textBox3.Text.Trim()) < estimatedDuration)
                {
                    locDeleted = 0;
                    locAdded = (estimatedDuration - Convert.ToDouble(textBox3.Text.Trim())) * locPerHour;
                }
                else
                {
                    locDeleted = 0;
                    locAdded = 0;
                }
            }
            else
            {
                locDeleted = Convert.ToDouble(textBox4.Text);
                locAdded = Convert.ToDouble(textBox5.Text);
            }
            Close();
        }
    }
}
```

```
namespace SDPTool
{
   partial class UpdateForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.changeTeamCheckBox = new System.Windows.Forms.CheckBox();
         this.addUserStoryCheckBox = new System.Windows.Forms.CheckBox();
         this.label1 = new System.Windows.Forms.Label();
         this.submitButton = new System.Windows.Forms.Button();
         this.changeUserStoryCheckBox = new System.Windows.Forms.CheckBox();
         this.removeUserStoryCheckBox = new System.Windows.Forms.CheckBox();
         this.updateStoryPriorityListCheckBox = new System.Windows.Forms.CheckBox();
         this.addTextBox = new System.Windows.Forms.TextBox();
         this.addLabel = new System.Windows.Forms.Label();
         this.removeLabel = new System.Windows.Forms.Label();
         this.removeTextBox = new System.Windows.Forms.TextBox();
         this.SuspendLayout();
         //
         // changeTeamCheckBox
         //
         this.changeTeamCheckBox.AutoSize = true;
         this.changeTeamCheckBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
         this.changeTeamCheckBox.Location = new System.Drawing.Point(25, 49);
         this.changeTeamCheckBox.Name = "changeTeamCheckBox";
         this.changeTeamCheckBox.Size = new System.Drawing.Size(135, 21);
```

```
            this.changeTeamCheckBox.TabIndex = 0;
            this.changeTeamCheckBox.Text = "Change Teams";
            this.changeTeamCheckBox.UseVisualStyleBackColor = true;
            //
            // addUserStoryCheckBox
            //
            this.addUserStoryCheckBox.AutoSize = true;
            this.addUserStoryCheckBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.addUserStoryCheckBox.Location = new System.Drawing.Point(25, 76);
            this.addUserStoryCheckBox.Name = "addUserStoryCheckBox";
            this.addUserStoryCheckBox.Size = new System.Drawing.Size(170, 21);
            this.addUserStoryCheckBox.TabIndex = 1;
            this.addUserStoryCheckBox.Text = "Add new User Story";
            this.addUserStoryCheckBox.UseVisualStyleBackColor = true;
            this.addUserStoryCheckBox.CheckedChanged += new
System.EventHandler(this.addUserStoryCheckBox_CheckedChanged);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(22, 20);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(345, 17);
            this.label1.TabIndex = 2;
            this.label1.Text = "Please check all the items you want to update:";
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(326, 174);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(75, 23);
            this.submitButton.TabIndex = 3;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // changeUserStoryCheckBox
            //
            this.changeUserStoryCheckBox.AutoSize = true;
            this.changeUserStoryCheckBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.changeUserStoryCheckBox.Location = new System.Drawing.Point(25, 130);
            this.changeUserStoryCheckBox.Name = "changeUserStoryCheckBox";
            this.changeUserStoryCheckBox.Size = new System.Drawing.Size(247, 21);
            this.changeUserStoryCheckBox.TabIndex = 4;
            this.changeUserStoryCheckBox.Text = "Change an existing User Story";
            this.changeUserStoryCheckBox.UseVisualStyleBackColor = true;
            //
```

```
            // removeUserStoryCheckBox
            //
            this.removeUserStoryCheckBox.AutoSize = true;
            this.removeUserStoryCheckBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeUserStoryCheckBox.Location = new System.Drawing.Point(25, 103);
            this.removeUserStoryCheckBox.Name = "removeUserStoryCheckBox";
            this.removeUserStoryCheckBox.Size = new System.Drawing.Size(167, 21);
            this.removeUserStoryCheckBox.TabIndex = 5;
            this.removeUserStoryCheckBox.Text = "Remove User Story";
            this.removeUserStoryCheckBox.UseVisualStyleBackColor = true;
            this.removeUserStoryCheckBox.CheckedChanged += new
System.EventHandler(this.removeUserStoryCheckBox_CheckedChanged);
            //
            // updateStoryPriorityListCheckBox
            //
            this.updateStoryPriorityListCheckBox.AutoSize = true;
            this.updateStoryPriorityListCheckBox.Font = new System.Drawing.Font("Microsoft Sans Serif",
10F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.updateStoryPriorityListCheckBox.Location = new System.Drawing.Point(25, 157);
            this.updateStoryPriorityListCheckBox.Name = "updateStoryPriorityListCheckBox";
            this.updateStoryPriorityListCheckBox.Size = new System.Drawing.Size(288, 21);
            this.updateStoryPriorityListCheckBox.TabIndex = 6;
            this.updateStoryPriorityListCheckBox.Text = "Update Story Priority List for Teams";
            this.updateStoryPriorityListCheckBox.UseVisualStyleBackColor = true;
            //
            // addTextBox
            //
            this.addTextBox.Location = new System.Drawing.Point(301, 76);
            this.addTextBox.Name = "addTextBox";
            this.addTextBox.Size = new System.Drawing.Size(100, 20);
            this.addTextBox.TabIndex = 7;
            this.addTextBox.Visible = false;
            //
            // addLabel
            //
            this.addLabel.AutoSize = true;
            this.addLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.addLabel.Location = new System.Drawing.Point(205, 77);
            this.addLabel.Name = "addLabel";
            this.addLabel.Size = new System.Drawing.Size(90, 17);
            this.addLabel.TabIndex = 8;
            this.addLabel.Text = "How many?";
            this.addLabel.Visible = false;
            //
            // removeLabel
            //
            this.removeLabel.AutoSize = true;
            this.removeLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeLabel.Location = new System.Drawing.Point(205, 104);
            this.removeLabel.Name = "removeLabel";
```

```
            this.removeLabel.Size = new System.Drawing.Size(90, 17);
            this.removeLabel.TabIndex = 10;
            this.removeLabel.Text = "How many?";
            this.removeLabel.Visible = false;
            //
            // removeTextBox
            //
            this.removeTextBox.Location = new System.Drawing.Point(301, 104);
            this.removeTextBox.Name = "removeTextBox";
            this.removeTextBox.Size = new System.Drawing.Size(100, 20);
            this.removeTextBox.TabIndex = 9;
            this.removeTextBox.Visible = false;
            //
            // UpdateForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(413, 209);
            this.Controls.Add(this.removeLabel);
            this.Controls.Add(this.removeTextBox);
            this.Controls.Add(this.addLabel);
            this.Controls.Add(this.addTextBox);
            this.Controls.Add(this.updateStoryPriorityListCheckBox);
            this.Controls.Add(this.removeUserStoryCheckBox);
            this.Controls.Add(this.changeUserStoryCheckBox);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.addUserStoryCheckBox);
            this.Controls.Add(this.changeTeamCheckBox);
            this.Name = "UpdateForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "UpdateForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.CheckBox changeTeamCheckBox;
        private System.Windows.Forms.CheckBox addUserStoryCheckBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.CheckBox changeUserStoryCheckBox;
        private System.Windows.Forms.CheckBox removeUserStoryCheckBox;
        private System.Windows.Forms.CheckBox updateStoryPriorityListCheckBox;
        private System.Windows.Forms.TextBox addTextBox;
        private System.Windows.Forms.Label addLabel;
        private System.Windows.Forms.Label removeLabel;
        private System.Windows.Forms.TextBox removeTextBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SDPTool
{
   public partial class UpdateForm : Form
   {
      bool changeTeam = false;
      bool addUserStory = false;
      int addCount = 0;
      bool removeUserStory = false;
      int removeCount = 0;
      bool changeUserStory = false;
      bool updateStoryPriorityList = false;

      public bool ChangeTeam
      {
         get  //get accessor method
         {
            return changeTeam;
         }
      }

      public bool AddUserStory
      {
         get  //get accessor method
         {
            return addUserStory;
         }
      }

      public int AddCount
      {
         get  //get accessor method
         {
            return addCount;
         }
      }

      public bool RemoveUserStory
      {
         get  //get accessor method
         {
            return removeUserStory;
         }
      }
```

```
public int RemoveCount
{
   get  //get accessor method
   {
      return removeCount;
   }
}

public bool ChangeUserStory
{
   get  //get accessor method
   {
      return changeUserStory;
   }
}

public bool UpdateStoryPriorityList
{
   get  //get accessor method
   {
      return updateStoryPriorityList;
   }
}

public UpdateForm()
{
   InitializeComponent();
}


private void submitButton_Click(object sender, EventArgs e)
{
   if (changeTeamCheckBox.Checked)
   {
      changeTeam = true;
   }
   if (addUserStoryCheckBox.Checked)
   {
      addUserStory = true;
      addCount = Int32.Parse(addTextBox.Text);
   }
   if (removeUserStoryCheckBox.Checked)
   {
      removeUserStory = true;
      removeCount = Int32.Parse(removeTextBox.Text);
   }
   if (changeUserStoryCheckBox.Checked)
   {
      changeUserStory = true;
   }
   if (updateStoryPriorityListCheckBox.Checked)
   {
      updateStoryPriorityList = true;
```

```
            }
            Close();
        }

        private void addUserStoryCheckBox_CheckedChanged(object sender, EventArgs e)
        {
            if (addUserStoryCheckBox.Checked)
            {
                addLabel.Visible = true;
                addTextBox.Visible = true;
            }
            else
            {
                addLabel.Visible = false;
                addTextBox.Visible = false;
            }
        }

        private void removeUserStoryCheckBox_CheckedChanged(object sender, EventArgs e)
        {
            if (removeUserStoryCheckBox.Checked)
            {
                removeLabel.Visible = true;
                removeTextBox.Visible = true;
            }
            else
            {
                removeLabel.Visible = false;
                removeTextBox.Visible = false;
            }
        }
    }
}
```

```
namespace SDPTool
{
   partial class UpdateTeamPriorityForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.label5 = new System.Windows.Forms.Label();
         this.teamTextBox = new System.Windows.Forms.TextBox();
         this.UpdateButton = new System.Windows.Forms.Button();
         this.label2 = new System.Windows.Forms.Label();
         this.currentStorylistBox = new System.Windows.Forms.ListBox();
         this.teamListBox = new System.Windows.Forms.ListBox();
         this.label1 = new System.Windows.Forms.Label();
         this.removeStoryButton = new System.Windows.Forms.Button();
         this.label8 = new System.Windows.Forms.Label();
         this.label9 = new System.Windows.Forms.Label();
         this.selectedStoryListBox = new System.Windows.Forms.ListBox();
         this.selectStoryButton = new System.Windows.Forms.Button();
         this.storyListBox = new System.Windows.Forms.ListBox();
         this.restartButton = new System.Windows.Forms.Button();
         this.exitButton = new System.Windows.Forms.Button();
         this.submitButton = new System.Windows.Forms.Button();
         this.finalizeButton = new System.Windows.Forms.Button();
         this.label4 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.teamSummaryBox = new System.Windows.Forms.RichTextBox();
         this.SuspendLayout();
```

```
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(163, 39);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(82, 13);
            this.label5.TabIndex = 80;
            this.label5.Text = "Selected Team:";
            //
            // teamTextBox
            //
            this.teamTextBox.Location = new System.Drawing.Point(166, 55);
            this.teamTextBox.Name = "teamTextBox";
            this.teamTextBox.Size = new System.Drawing.Size(118, 20);
            this.teamTextBox.TabIndex = 81;
            //
            // UpdateButton
            //
            this.UpdateButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.UpdateButton.Location = new System.Drawing.Point(36, 214);
            this.UpdateButton.Name = "UpdateButton";
            this.UpdateButton.Size = new System.Drawing.Size(95, 29);
            this.UpdateButton.TabIndex = 79;
            this.UpdateButton.Text = "Update";
            this.UpdateButton.UseVisualStyleBackColor = true;
            this.UpdateButton.Click += new System.EventHandler(this.UpdateButton_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(163, 81);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(79, 13);
            this.label2.TabIndex = 77;
            this.label2.Text = "Current Stories:";
            //
            // currentStorylistBox
            //
            this.currentStorylistBox.FormattingEnabled = true;
            this.currentStorylistBox.Location = new System.Drawing.Point(166, 94);
            this.currentStorylistBox.Name = "currentStorylistBox";
            this.currentStorylistBox.Size = new System.Drawing.Size(118, 108);
            this.currentStorylistBox.TabIndex = 78;
            //
            // teamListBox
            //
            this.teamListBox.FormattingEnabled = true;
            this.teamListBox.Location = new System.Drawing.Point(22, 55);
            this.teamListBox.Name = "teamListBox";
            this.teamListBox.Size = new System.Drawing.Size(118, 147);
            this.teamListBox.TabIndex = 76;
```

```
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(19, 39);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(42, 13);
            this.label1.TabIndex = 75;
            this.label1.Text = "Teams:";
            //
            // removeStoryButton
            //
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(439, 107);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 72;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(481, 39);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(112, 13);
            this.label8.TabIndex = 73;
            this.label8.Text = "User Story Priority List:";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(307, 39);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(67, 13);
            this.label9.TabIndex = 69;
            this.label9.Text = "User Stories:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(484, 55);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 121);
            this.selectedStoryListBox.TabIndex = 74;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(439, 63);
```

```
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 71;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(310, 55);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 121);
            this.storyListBox.TabIndex = 70;
            //
            // restartButton
            //
            this.restartButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.restartButton.Location = new System.Drawing.Point(769, 42);
            this.restartButton.Name = "restartButton";
            this.restartButton.Size = new System.Drawing.Size(95, 29);
            this.restartButton.TabIndex = 68;
            this.restartButton.Text = "Restart";
            this.restartButton.UseVisualStyleBackColor = true;
            this.restartButton.Click += new System.EventHandler(this.restartButton_Click);
            //
            // exitButton
            //
            this.exitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.exitButton.Location = new System.Drawing.Point(769, 225);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(95, 29);
            this.exitButton.TabIndex = 67;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(507, 187);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 63;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // finalizeButton
            //
```

```
        this.finalizeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.finalizeButton.Location = new System.Drawing.Point(668, 225);
        this.finalizeButton.Name = "finalizeButton";
        this.finalizeButton.Size = new System.Drawing.Size(95, 29);
        this.finalizeButton.TabIndex = 66;
        this.finalizeButton.Text = "Finalize";
        this.finalizeButton.UseVisualStyleBackColor = true;
        this.finalizeButton.Click += new System.EventHandler(this.finalizeButton_Click);
        //
        // label4
        //
        this.label4.AutoSize = true;
        this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.label4.Location = new System.Drawing.Point(19, 13);
        this.label4.Name = "label4";
        this.label4.Size = new System.Drawing.Size(341, 17);
        this.label4.TabIndex = 60;
        this.label4.Text = "Please update the user stories for each team:";
        //
        // label3
        //
        this.label3.AutoSize = true;
        this.label3.Location = new System.Drawing.Point(611, 26);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(42, 13);
        this.label3.TabIndex = 64;
        this.label3.Text = "Teams:";
        //
        // teamSummaryBox
        //
        this.teamSummaryBox.Location = new System.Drawing.Point(614, 42);
        this.teamSummaryBox.Name = "teamSummaryBox";
        this.teamSummaryBox.ReadOnly = true;
        this.teamSummaryBox.Size = new System.Drawing.Size(149, 177);
        this.teamSummaryBox.TabIndex = 65;
        this.teamSummaryBox.Text = "";
        //
        // UpdateTeamPriorityForm
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(890, 266);
        this.Controls.Add(this.label5);
        this.Controls.Add(this.teamTextBox);
        this.Controls.Add(this.UpdateButton);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.currentStorylistBox);
        this.Controls.Add(this.teamListBox);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.removeStoryButton);
        this.Controls.Add(this.label8);
```

```
            this.Controls.Add(this.label9);
            this.Controls.Add(this.selectedStoryListBox);
            this.Controls.Add(this.selectStoryButton);
            this.Controls.Add(this.storyListBox);
            this.Controls.Add(this.restartButton);
            this.Controls.Add(this.exitButton);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.finalizeButton);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.teamSummaryBox);
            this.Name = "UpdateTeamPriorityForm";
            this.Text = "UpdateTeamPriorityForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox teamTextBox;
        private System.Windows.Forms.Button UpdateButton;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox currentStorylistBox;
        private System.Windows.Forms.ListBox teamListBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button removeStoryButton;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.ListBox selectedStoryListBox;
        private System.Windows.Forms.Button selectStoryButton;
        private System.Windows.Forms.ListBox storyListBox;
        private System.Windows.Forms.Button restartButton;
        private System.Windows.Forms.Button exitButton;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Button finalizeButton;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.RichTextBox teamSummaryBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class UpdateTeamPriorityForm : Form
   {

      static int MAX_STORY = 50;
      FileStream teamFile = null;
      StreamWriter sw = null;
      string fileName = "updateteams.txt";
      int totalTeams = 0;
      int totalStory = 0;
      string[] storyName = new string[MAX_STORY];
      int storyCount = -1;


      public UpdateTeamPriorityForm(int tCount, int sCount)
      {
         InitializeComponent();
         UpdateButton.Enabled = true;
         storyListBox.Enabled = false;
         submitButton.Enabled = false;
         selectStoryButton.Enabled = true;
         removeStoryButton.Enabled = false;
         finalizeButton.Enabled = true;
         try
         {
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
         }
         catch (Exception e)
         {
            Console.WriteLine(e.ToString());
            MessageBox.Show("File team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
         }
         totalTeams = tCount;
         totalStory = sCount;
      }

      public int TotalTeams
      {
         get
         {
```

```
            return totalTeams;
        }
        set
        {
            totalTeams = value;
        }
    }

    public void AddTeam(string team)
    {
        teamListBox.Items.Add(team);
    }

    public void AddStory(string story)
    {
        storyListBox.Items.Add(story);
        storyName[++storyCount] = story;
    }

    private void restartButton_Click(object sender, EventArgs e)
    {

        storyListBox.Items.Clear();
        selectedStoryListBox.Items.Clear();

        teamSummaryBox.Text = null;

        submitButton.Enabled = false;

        try
        {
            if (teamFile != null)
                teamFile.Close();
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.ToString());
            MessageBox.Show("File: team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }

        for (int i = 0; i <= storyCount; i++)
        {
            storyListBox.Items.Add(storyName[i]);
        }
    }

    private void finalizeButton_Click(object sender, EventArgs e)
    {
        if (teamFile != null)
        {
```

```
            sw.Close();
            teamFile.Close();
        }
        Close();
    }

    private void submitButton_Click(object sender, EventArgs e)
    {
        teamSummaryBox.AppendText(string.Concat("Team: ", teamTextBox.Text.Trim(), ":\n"));
        sw.Write(teamTextBox.Text.Trim());

        int sCount = 0;
        foreach (object item in selectedStoryListBox.Items)
        {
            teamSummaryBox.AppendText(string.Concat("UserStory  ", Convert.ToString(++sCount), ": ",
(string)item, "\n"));
            sw.Write(string.Concat(", ", (string)item));
        }


        selectedStoryListBox.Items.Clear();

        teamTextBox.Text = null;
        submitButton.Enabled = false;
        storyListBox.Enabled = false;
        selectedStoryListBox.Enabled = false;
        selectStoryButton.Enabled = false;
        removeStoryButton.Enabled = false;
        UpdateButton.Enabled = true;
        teamSummaryBox.ScrollToCaret();
    }


    private void exitButton_Click(object sender, EventArgs e)
    {
        this.Close();
        Environment.Exit(0);
    }

    private void selectStoryButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)storyListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedStoryListBox.Items.Add(selectedText);
            storyListBox.Items.Remove(storyListBox.SelectedItem);
            removeStoryButton.Enabled = true;
        }
        else
        {
            MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
```

```
        }

        private void removeStoryButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedStoryListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedStoryListBox.Items.Remove(selectedItem);
                storyListBox.Items.Add(selectedItem);

                if (selectedStoryListBox.Items.Count == 0)
                    removeStoryButton.Enabled = false;
            }
        }

        private void UpdateButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)teamListBox.SelectedItem;
            if (selectedItem != null)
            {
                currentStorylistBox.Items.Clear();
                storyListBox.Enabled = true;
                submitButton.Enabled = true;
                selectStoryButton.Enabled = true;
                removeStoryButton.Enabled = false;
                teamTextBox.Text = selectedItem;
                UpdateButton.Enabled = false;
            }
        }
    }
}
```

```
namespace SDPTool
{
   partial class TeamForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.teamSummaryBox = new System.Windows.Forms.RichTextBox();
         this.agentListBox = new System.Windows.Forms.ListBox();
         this.selectButton = new System.Windows.Forms.Button();
         this.createTeamButton = new System.Windows.Forms.Button();
         this.label1 = new System.Windows.Forms.Label();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.selectedAgentListBox = new System.Windows.Forms.ListBox();
         this.label4 = new System.Windows.Forms.Label();
         this.finalizeButton = new System.Windows.Forms.Button();
         this.removeButton = new System.Windows.Forms.Button();
         this.label5 = new System.Windows.Forms.Label();
         this.submitButton = new System.Windows.Forms.Button();
         this.activityComboBox = new System.Windows.Forms.ComboBox();
         this.label6 = new System.Windows.Forms.Label();
         this.storyCountTextBox = new System.Windows.Forms.TextBox();
         this.label7 = new System.Windows.Forms.Label();
         this.teamMembersTextBox = new System.Windows.Forms.RichTextBox();
         this.exitButton = new System.Windows.Forms.Button();
         this.restartButton = new System.Windows.Forms.Button();
         this.removeStoryButton = new System.Windows.Forms.Button();
```

```
            this.label8 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.selectedStoryListBox = new System.Windows.Forms.ListBox();
            this.selectStoryButton = new System.Windows.Forms.Button();
            this.storyListBox = new System.Windows.Forms.ListBox();
            this.SuspendLayout();
            //
            // teamSummaryBox
            //
            this.teamSummaryBox.Location = new System.Drawing.Point(624, 38);
            this.teamSummaryBox.Name = "teamSummaryBox";
            this.teamSummaryBox.ReadOnly = true;
            this.teamSummaryBox.Size = new System.Drawing.Size(149, 177);
            this.teamSummaryBox.TabIndex = 16;
            this.teamSummaryBox.Text = "";
            //
            // agentListBox
            //
            this.agentListBox.FormattingEnabled = true;
            this.agentListBox.Location = new System.Drawing.Point(12, 68);
            this.agentListBox.Name = "agentListBox";
            this.agentListBox.Size = new System.Drawing.Size(118, 147);
            this.agentListBox.TabIndex = 2;
            //
            // selectButton
            //
            this.selectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectButton.Location = new System.Drawing.Point(136, 83);
            this.selectButton.Name = "selectButton";
            this.selectButton.Size = new System.Drawing.Size(39, 38);
            this.selectButton.TabIndex = 3;
            this.selectButton.Text = ">>";
            this.selectButton.UseVisualStyleBackColor = true;
            this.selectButton.Click += new System.EventHandler(this.selectButton_Click);
            //
            // createTeamButton
            //
            this.createTeamButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.createTeamButton.Location = new System.Drawing.Point(189, 186);
            this.createTeamButton.Name = "createTeamButton";
            this.createTeamButton.Size = new System.Drawing.Size(95, 29);
            this.createTeamButton.TabIndex = 7;
            this.createTeamButton.Text = "Create Team";
            this.createTeamButton.UseVisualStyleBackColor = true;
            this.createTeamButton.Click += new System.EventHandler(this.createTeamButton_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(12, 52);
            this.label1.Name = "label1";
```

```
            this.label1.Size = new System.Drawing.Size(75, 13);
            this.label1.TabIndex = 1;
            this.label1.Text = "Employee List:";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(178, 52);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(106, 13);
            this.label2.TabIndex = 5;
            this.label2.Text = "Selected Employees:";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(621, 22);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(42, 13);
            this.label3.TabIndex = 15;
            this.label3.Text = "Teams:";
            //
            // selectedAgentListBox
            //
            this.selectedAgentListBox.FormattingEnabled = true;
            this.selectedAgentListBox.Location = new System.Drawing.Point(181, 68);
            this.selectedAgentListBox.Name = "selectedAgentListBox";
            this.selectedAgentListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedAgentListBox.TabIndex = 6;
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(9, 9);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(331, 17);
            this.label4.TabIndex = 0;
            this.label4.Text = "Please create teams of two employees each:";
            //
            // finalizeButton
            //
            this.finalizeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.finalizeButton.Location = new System.Drawing.Point(780, 178);
            this.finalizeButton.Name = "finalizeButton";
            this.finalizeButton.Size = new System.Drawing.Size(95, 29);
            this.finalizeButton.TabIndex = 17;
            this.finalizeButton.Text = "Finalize";
            this.finalizeButton.UseVisualStyleBackColor = true;
            this.finalizeButton.Click += new System.EventHandler(this.finalizeButton_Click);
            //
```

```
            // removeButton
            //
            this.removeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeButton.Location = new System.Drawing.Point(136, 137);
            this.removeButton.Name = "removeButton";
            this.removeButton.Size = new System.Drawing.Size(39, 38);
            this.removeButton.TabIndex = 4;
            this.removeButton.Text = "<<";
            this.removeButton.UseVisualStyleBackColor = true;
            this.removeButton.Click += new System.EventHandler(this.removeButton_Click);
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(471, 22);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(83, 13);
            this.label5.TabIndex = 8;
            this.label5.Text = "Team Members:";
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(493, 165);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 14;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // activityComboBox
            //
            this.activityComboBox.FormattingEnabled = true;
            this.activityComboBox.Location = new System.Drawing.Point(437, 213);
            this.activityComboBox.Name = "activityComboBox";
            this.activityComboBox.Size = new System.Drawing.Size(118, 21);
            this.activityComboBox.TabIndex = 11;
            this.activityComboBox.Text = "Planning";
            this.activityComboBox.Visible = false;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(434, 197);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(79, 13);
            this.label6.TabIndex = 10;
            this.label6.Text = "Alloted Activity:";
            this.label6.Visible = false;
            //
```

```
            // storyCountTextBox
            //
            this.storyCountTextBox.Location = new System.Drawing.Point(308, 60);
            this.storyCountTextBox.Name = "storyCountTextBox";
            this.storyCountTextBox.Size = new System.Drawing.Size(118, 20);
            this.storyCountTextBox.TabIndex = 13;
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(305, 44);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(104, 13);
            this.label7.TabIndex = 12;
            this.label7.Text = "Total Alloted Stories:";
            //
            // teamMembersTextBox
            //
            this.teamMembersTextBox.Location = new System.Drawing.Point(482, 38);
            this.teamMembersTextBox.Name = "teamMembersTextBox";
            this.teamMembersTextBox.ReadOnly = true;
            this.teamMembersTextBox.ScrollBars = System.Windows.Forms.RichTextBoxScrollBars.None;
            this.teamMembersTextBox.Size = new System.Drawing.Size(118, 32);
            this.teamMembersTextBox.TabIndex = 9;
            this.teamMembersTextBox.Text = "";
            //
            // exitButton
            //
            this.exitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.exitButton.Location = new System.Drawing.Point(780, 213);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(95, 29);
            this.exitButton.TabIndex = 18;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // restartButton
            //
            this.restartButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.restartButton.Location = new System.Drawing.Point(780, 38);
            this.restartButton.Name = "restartButton";
            this.restartButton.Size = new System.Drawing.Size(95, 29);
            this.restartButton.TabIndex = 19;
            this.restartButton.Text = "Restart";
            this.restartButton.UseVisualStyleBackColor = true;
            this.restartButton.Click += new System.EventHandler(this.restartButton_Click);
            //
            // removeStoryButton
            //
```

```
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(437, 134);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 23;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(479, 74);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(87, 13);
            this.label8.TabIndex = 24;
            this.label8.Text = "Selected Stories:";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(305, 83);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(67, 13);
            this.label9.TabIndex = 20;
            this.label9.Text = "User Stories:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(482, 90);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 69);
            this.selectedStoryListBox.TabIndex = 25;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(437, 90);
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 22;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(308, 99);
            this.storyListBox.Name = "storyListBox";
```

```
            this.storyListBox.Size = new System.Drawing.Size(118, 108);
            this.storyListBox.TabIndex = 21;
            //
            // TeamForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(886, 251);
            this.ControlBox = false;
            this.Controls.Add(this.removeStoryButton);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.selectedStoryListBox);
            this.Controls.Add(this.selectStoryButton);
            this.Controls.Add(this.storyListBox);
            this.Controls.Add(this.restartButton);
            this.Controls.Add(this.exitButton);
            this.Controls.Add(this.teamMembersTextBox);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.storyCountTextBox);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.activityComboBox);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.removeButton);
            this.Controls.Add(this.finalizeButton);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.createTeamButton);
            this.Controls.Add(this.selectedAgentListBox);
            this.Controls.Add(this.selectButton);
            this.Controls.Add(this.agentListBox);
            this.Controls.Add(this.teamSummaryBox);
            this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "TeamForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Creating Teams ...";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.RichTextBox teamSummaryBox;
        private System.Windows.Forms.ListBox agentListBox;
        private System.Windows.Forms.Button selectButton;
        private System.Windows.Forms.Button createTeamButton;
```

```
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.ListBox selectedAgentListBox;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Button finalizeButton;
        private System.Windows.Forms.Button removeButton;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.ComboBox activityComboBox;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox storyCountTextBox;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.RichTextBox teamMembersTextBox;
        private System.Windows.Forms.Button exitButton;
        private System.Windows.Forms.Button restartButton;
        private System.Windows.Forms.Button removeStoryButton;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.ListBox selectedStoryListBox;
        private System.Windows.Forms.Button selectStoryButton;
        private System.Windows.Forms.ListBox storyListBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class TeamForm : Form
   {
      static int MAX_AGENTS = 50;
      static int MAX_STORY = 50;
      int teamCount = 0;
      string folderPath = "C:\\SDPTool\\Results\\";
      FileStream teamFile;
      StreamWriter sw;
      int totalTeams = 0;
      int totalStory = 0;
      string[] agentName = new string[MAX_AGENTS];
      int agentCount = -1;
      string[] storyName = new string[MAX_STORY];
      int storyCount = -1;

      public string FolderPath
      {
         get  //get accessor method
         {
            return folderPath;
         }
         set   //set accessor method
         {
            folderPath = value;
         }
      }

      public TeamForm(int tCount, int sCount)
      {
         InitializeComponent();
         removeButton.Enabled = false;
         createTeamButton.Enabled = false;

         teamMembersTextBox.Text = null;
         activityComboBox.Enabled = false;
         storyCountTextBox.Text = "0";
         storyCountTextBox.Enabled = false;
         storyListBox.Enabled = false;
         selectedStoryListBox.Enabled = false;
         submitButton.Enabled = false;
         selectStoryButton.Enabled = false;
```

```
            removeStoryButton.Enabled = false;
            try
            {
                teamFile = new FileStream(folderPath + "teams.txt", FileMode.Create, FileAccess.Write);
                sw = new StreamWriter(teamFile);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
                MessageBox.Show("File team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }

            totalTeams = tCount;
            totalStory = sCount;
            if (totalTeams > 0)
            {
                finalizeButton.Enabled = false;
            }
        }



        public int TotalTeams
        {
            get
            {
                return totalTeams;
            }
            set
            {
                totalTeams = value;
            }
        }

        public void AddStory(string story)
        {
            storyListBox.Items.Add(story);
            storyName[++storyCount] = story;
        }


        public void AddAgent(string agent)
        {
            agentListBox.Items.Add(agent);
            agentName[++agentCount] = agent;
        }

        private void removeButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedAgentListBox.SelectedItem;
            if (selectedItem != null)
            {
```

```
            selectedAgentListBox.Items.Remove(selectedItem);
            agentListBox.Items.Add(selectedItem);

            if (selectedAgentListBox.Items.Count == 0)
                removeButton.Enabled = false;
            if (selectedAgentListBox.Items.Count == 2)
                createTeamButton.Enabled = true;
            else
                createTeamButton.Enabled = false;
        }
    }

    private void selectButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)agentListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedAgentListBox.Items.Add(selectedText);
            agentListBox.Items.Remove(agentListBox.SelectedItem);

            removeButton.Enabled = true;
            if (selectedAgentListBox.Items.Count == 2)
                createTeamButton.Enabled = true;
            else
                createTeamButton.Enabled = false;
        }
        else
        {
            MessageBox.Show("Please Select an Employee to start creating a team", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void createTeamButton_Click(object sender, EventArgs e)
    {
        teamMembersTextBox.Text = null;
        int count = 0;
        foreach (object item in selectedAgentListBox.Items)
        {
            if (++count != 2)
                teamMembersTextBox.AppendText(string.Concat((string)item, "\n"));
            else
                teamMembersTextBox.AppendText((string)item);
        }
        teamMembersTextBox.ScrollToCaret();
        DisableAllButtons();
        activityComboBox.Enabled = true;
        storyCountTextBox.Enabled = true;
        storyCountTextBox.Text = "0";
        storyListBox.Enabled = true;
        selectedStoryListBox.Enabled = true;
        submitButton.Enabled = true;
        selectStoryButton.Enabled = true;
```

```
            if (teamCount >= totalTeams)
            {
                finalizeButton.Enabled = true;
            }
        }

        private void restartButton_Click(object sender, EventArgs e)
        {
            agentListBox.Items.Clear();
            selectedAgentListBox.Items.Clear();
            storyListBox.Items.Clear();
            selectedStoryListBox.Items.Clear();
            teamMembersTextBox.Text = null;
            teamSummaryBox.Text = null;

            selectButton.Enabled = true;
            removeButton.Enabled = false;
            createTeamButton.Enabled = false;

            activityComboBox.Enabled = false;
            storyCountTextBox.Text = "0";
            storyCountTextBox.Enabled = false;
            submitButton.Enabled = false;

            teamCount = 0;

            try
            {
                if (teamFile != null)
                    teamFile.Close();
                teamFile = new FileStream(folderPath + "teams.txt", FileMode.Create, FileAccess.Write);
                sw = new StreamWriter(teamFile);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
                MessageBox.Show("File: team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }

            for (int i = 0; i <= agentCount; i++)
            {
                agentListBox.Items.Add(agentName[i]);
            }

            for (int i = 0; i <= storyCount; i++)
            {
                storyListBox.Items.Add(storyName[i]);
            }
        }

        private void finalizeButton_Click(object sender, EventArgs e)
        {
```

```
            if (teamFile != null)
            {
                sw.Close();
                teamFile.Close();
            }
            Close();
        }

        private void submitButton_Click(object sender, EventArgs e)
        {
            teamSummaryBox.AppendText(string.Concat("Team", Convert.ToString(++teamCount), ":\n"));
            sw.Write(string.Concat("Team", Convert.ToString(teamCount)));
            int tCount = 0;
            foreach (object item in selectedAgentListBox.Items)
            {
                teamSummaryBox.AppendText(string.Concat("Member ", Convert.ToString(++tCount), ": ",
(string)item, "\n"));
                sw.Write(string.Concat(", ", (string)item));
            }
            int sCount = 0;
            foreach (object item in selectedStoryListBox.Items)
            {
                teamSummaryBox.AppendText(string.Concat("UserStory  ", Convert.ToString(++sCount), ": ",
(string)item, "\n"));
                sw.Write(string.Concat(", ", (string)item));
            }

            teamSummaryBox.AppendText(string.Concat("Alloted # of Stories: ", storyCountTextBox.Text,
"\n\n"));

            sw.Write(string.Concat(", ", storyCountTextBox.Text));
            sw.Write("\n");
            teamMembersTextBox.Text = null;
            selectedAgentListBox.Items.Clear();
            selectedStoryListBox.Items.Clear();
            activityComboBox.Text = "Planning";
            activityComboBox.Enabled = false;
            storyCountTextBox.Text = "0";
            storyCountTextBox.Enabled = false;
            selectButton.Enabled = true;
            submitButton.Enabled = false;
            storyListBox.Enabled = false;
            selectedStoryListBox.Enabled = false;
            selectStoryButton.Enabled = false;
            removeStoryButton.Enabled = false;
            if (teamCount >= totalTeams)
            {
                finalizeButton.Enabled = true;
            }
            teamSummaryBox.ScrollToCaret();
        }

        private void EnableAllButtons()
```

```
        {
            removeButton.Enabled = true;
            selectButton.Enabled = true;
        }

        private void DisableAllButtons()
        {
            createTeamButton.Enabled = false;
            removeButton.Enabled = false;
            selectButton.Enabled = false;
        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
            Environment.Exit(0);
        }

        private void selectStoryButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
                selectedStoryListBox.Items.Add(selectedText);
                storyListBox.Items.Remove(storyListBox.SelectedItem);
                removeStoryButton.Enabled = true;
                storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) + 1);
            }
            else
            {
                MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void removeStoryButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedStoryListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedStoryListBox.Items.Remove(selectedItem);
                storyListBox.Items.Add(selectedItem);
                storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) - 1);

                if (selectedStoryListBox.Items.Count == 0)
                    removeStoryButton.Enabled = false;
            }
        }

    }
}
```

```
namespace SDPTool
{
   partial class TeamUpdateForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.removeStoryButton = new System.Windows.Forms.Button();
         this.label8 = new System.Windows.Forms.Label();
         this.label9 = new System.Windows.Forms.Label();
         this.selectedStoryListBox = new System.Windows.Forms.ListBox();
         this.selectStoryButton = new System.Windows.Forms.Button();
         this.storyListBox = new System.Windows.Forms.ListBox();
         this.restartButton = new System.Windows.Forms.Button();
         this.exitButton = new System.Windows.Forms.Button();
         this.label7 = new System.Windows.Forms.Label();
         this.storyCountTextBox = new System.Windows.Forms.TextBox();
         this.submitButton = new System.Windows.Forms.Button();
         this.finalizeButton = new System.Windows.Forms.Button();
         this.label4 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
         this.teamSummaryBox = new System.Windows.Forms.RichTextBox();
         this.label1 = new System.Windows.Forms.Label();
         this.teamListBox = new System.Windows.Forms.ListBox();
         this.label2 = new System.Windows.Forms.Label();
         this.currentStorylistBox = new System.Windows.Forms.ListBox();
         this.UpdateButton = new System.Windows.Forms.Button();
         this.label5 = new System.Windows.Forms.Label();
```

```
            this.teamTextBox = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // removeStoryButton
            //
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(438, 107);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 49;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(480, 39);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(87, 13);
            this.label8.TabIndex = 50;
            this.label8.Text = "Selected Stories:";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(306, 39);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(67, 13);
            this.label9.TabIndex = 46;
            this.label9.Text = "User Stories:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(483, 55);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 121);
            this.selectedStoryListBox.TabIndex = 51;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(438, 63);
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 48;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
```

```
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(309, 55);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 121);
            this.storyListBox.TabIndex = 47;
            //
            // restartButton
            //
            this.restartButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.restartButton.Location = new System.Drawing.Point(768, 42);
            this.restartButton.Name = "restartButton";
            this.restartButton.Size = new System.Drawing.Size(95, 29);
            this.restartButton.TabIndex = 45;
            this.restartButton.Text = "Restart";
            this.restartButton.UseVisualStyleBackColor = true;
            this.restartButton.Click += new System.EventHandler(this.restartButton_Click);
            //
            // exitButton
            //
            this.exitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.exitButton.Location = new System.Drawing.Point(768, 225);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(95, 29);
            this.exitButton.TabIndex = 44;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(162, 209);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(104, 13);
            this.label7.TabIndex = 38;
            this.label7.Text = "Total Alloted Stories:";
            //
            // storyCountTextBox
            //
            this.storyCountTextBox.Location = new System.Drawing.Point(165, 225);
            this.storyCountTextBox.Name = "storyCountTextBox";
            this.storyCountTextBox.Size = new System.Drawing.Size(118, 20);
            this.storyCountTextBox.TabIndex = 39;
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(506, 187);
```

```
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 40;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // finalizeButton
            //
            this.finalizeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.finalizeButton.Location = new System.Drawing.Point(667, 225);
            this.finalizeButton.Name = "finalizeButton";
            this.finalizeButton.Size = new System.Drawing.Size(95, 29);
            this.finalizeButton.TabIndex = 43;
            this.finalizeButton.Text = "Finalize";
            this.finalizeButton.UseVisualStyleBackColor = true;
            this.finalizeButton.Click += new System.EventHandler(this.finalizeButton_Click);
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(18, 13);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(341, 17);
            this.label4.TabIndex = 26;
            this.label4.Text = "Please update the user stories for each team:";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(610, 26);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(42, 13);
            this.label3.TabIndex = 41;
            this.label3.Text = "Teams:";
            //
            // teamSummaryBox
            //
            this.teamSummaryBox.Location = new System.Drawing.Point(613, 42);
            this.teamSummaryBox.Name = "teamSummaryBox";
            this.teamSummaryBox.ReadOnly = true;
            this.teamSummaryBox.Size = new System.Drawing.Size(149, 177);
            this.teamSummaryBox.TabIndex = 42;
            this.teamSummaryBox.Text = "";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(18, 39);
            this.label1.Name = "label1";
```

```
            this.label1.Size = new System.Drawing.Size(42, 13);
            this.label1.TabIndex = 53;
            this.label1.Text = "Teams:";
            //
            // teamListBox
            //
            this.teamListBox.FormattingEnabled = true;
            this.teamListBox.Location = new System.Drawing.Point(21, 55);
            this.teamListBox.Name = "teamListBox";
            this.teamListBox.Size = new System.Drawing.Size(118, 147);
            this.teamListBox.TabIndex = 54;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(162, 81);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(79, 13);
            this.label2.TabIndex = 55;
            this.label2.Text = "Current Stories:";
            //
            // currentStorylistBox
            //
            this.currentStorylistBox.FormattingEnabled = true;
            this.currentStorylistBox.Location = new System.Drawing.Point(165, 94);
            this.currentStorylistBox.Name = "currentStorylistBox";
            this.currentStorylistBox.Size = new System.Drawing.Size(118, 108);
            this.currentStorylistBox.TabIndex = 56;
            //
            // UpdateButton
            //
            this.UpdateButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.UpdateButton.Location = new System.Drawing.Point(35, 214);
            this.UpdateButton.Name = "UpdateButton";
            this.UpdateButton.Size = new System.Drawing.Size(95, 29);
            this.UpdateButton.TabIndex = 57;
            this.UpdateButton.Text = "Update";
            this.UpdateButton.UseVisualStyleBackColor = true;
            this.UpdateButton.Click += new System.EventHandler(this.UpdateButton_Click);
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(162, 39);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(82, 13);
            this.label5.TabIndex = 58;
            this.label5.Text = "Selected Team:";
            //
            // teamTextBox
            //
            this.teamTextBox.Location = new System.Drawing.Point(165, 55);
```

```
    this.teamTextBox.Name = "teamTextBox";
    this.teamTextBox.Size = new System.Drawing.Size(118, 20);
    this.teamTextBox.TabIndex = 59;
    //
    // TeamUpdateForm
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(881, 266);
    this.Controls.Add(this.label5);
    this.Controls.Add(this.teamTextBox);
    this.Controls.Add(this.UpdateButton);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.currentStorylistBox);
    this.Controls.Add(this.teamListBox);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.removeStoryButton);
    this.Controls.Add(this.label8);
    this.Controls.Add(this.label9);
    this.Controls.Add(this.selectedStoryListBox);
    this.Controls.Add(this.selectStoryButton);
    this.Controls.Add(this.storyListBox);
    this.Controls.Add(this.restartButton);
    this.Controls.Add(this.exitButton);
    this.Controls.Add(this.label7);
    this.Controls.Add(this.storyCountTextBox);
    this.Controls.Add(this.submitButton);
    this.Controls.Add(this.finalizeButton);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.teamSummaryBox);
    this.Name = "TeamUpdateForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
    this.Text = "TeamUpdateForm";
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button removeStoryButton;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.ListBox selectedStoryListBox;
private System.Windows.Forms.Button selectStoryButton;
private System.Windows.Forms.ListBox storyListBox;
private System.Windows.Forms.Button restartButton;
private System.Windows.Forms.Button exitButton;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.TextBox storyCountTextBox;
private System.Windows.Forms.Button submitButton;
private System.Windows.Forms.Button finalizeButton;
```

```
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.RichTextBox teamSummaryBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ListBox teamListBox;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox currentStorylistBox;
        private System.Windows.Forms.Button UpdateButton;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox teamTextBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class TeamUpdateForm : Form
   {
      static int MAX_STORY = 50;
      FileStream teamFile = null;
      StreamWriter sw = null;
      string fileName = "updateteams.txt";
      int totalTeams = 0;
      int totalStory = 0;
      string[] storyName = new string[MAX_STORY];
      int storyCount = -1;


      public TeamUpdateForm(int tCount, int sCount)
      {
         InitializeComponent();
         UpdateButton.Enabled = true;
         storyCountTextBox.Text = "0";
         storyCountTextBox.Enabled = false;
         storyListBox.Enabled = false;
         submitButton.Enabled = false;
         selectStoryButton.Enabled = true;
         removeStoryButton.Enabled = false;
         finalizeButton.Enabled = true;
         try
         {
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
         }
         catch (Exception e)
         {
            Console.WriteLine(e.ToString());
            MessageBox.Show("File team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
         }
         totalTeams = tCount;
         totalStory = sCount;
      }

      public int TotalTeams
      {
         get
```

```
        {
            return totalTeams;
        }
        set
        {
            totalTeams = value;
        }
    }

    public void AddTeam(string team)
    {
        teamListBox.Items.Add(team);
    }

    public void AddStory(string story)
    {
        storyListBox.Items.Add(story);
        storyName[++storyCount] = story;
    }

    private void restartButton_Click(object sender, EventArgs e)
    {

        storyListBox.Items.Clear();
        selectedStoryListBox.Items.Clear();

        teamSummaryBox.Text = null;

        storyCountTextBox.Text = "0";
        storyCountTextBox.Enabled = false;
        submitButton.Enabled = false;

        try
        {
            if (teamFile != null)
                teamFile.Close();
            teamFile = new FileStream(fileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(teamFile);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.ToString());
            MessageBox.Show("File: team.txt already in use", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }


        for (int i = 0; i <= storyCount; i++)
        {
            storyListBox.Items.Add(storyName[i]);
        }
    }
```

```
        private void finalizeButton_Click(object sender, EventArgs e)
        {
            if (teamFile != null)
            {
                sw.Close();
                teamFile.Close();
            }
            Close();
        }

        private void submitButton_Click(object sender, EventArgs e)
        {
            teamSummaryBox.AppendText(string.Concat("Team: ", teamTextBox.Text.Trim(), ":\n"));
            sw.Write(teamTextBox.Text.Trim());

            int sCount = 0;
            foreach (object item in selectedStoryListBox.Items)
            {
                teamSummaryBox.AppendText(string.Concat("UserStory  ", Convert.ToString(++sCount), ": ",
(string)item, "\n"));
                sw.Write(string.Concat(", ", (string)item));
            }

            teamSummaryBox.AppendText(string.Concat("Alloted # of Stories: ", storyCountTextBox.Text,
"\n\n"));
            sw.Write(string.Concat(", ", storyCountTextBox.Text));
            sw.Write("\n");

            selectedStoryListBox.Items.Clear();
            storyCountTextBox.Text = "0";
            storyCountTextBox.Enabled = false;
            teamTextBox.Text = null;
            submitButton.Enabled = false;
            storyListBox.Enabled = false;
            selectedStoryListBox.Enabled = false;
            selectStoryButton.Enabled = false;
            removeStoryButton.Enabled = false;
            UpdateButton.Enabled = true;
            teamSummaryBox.ScrollToCaret();
        }


        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
            Environment.Exit(0);
        }

        private void selectStoryButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
```

```
            //Agent selectedAgent = org.GetAgent(selectedText);
            selectedStoryListBox.Items.Add(selectedText);
            //agentListBox.ClearSelected();
            storyListBox.Items.Remove(storyListBox.SelectedItem);
            removeStoryButton.Enabled = true;
            storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) + 1);
        }
        else
        {
            MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void removeStoryButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)selectedStoryListBox.SelectedItem;
        if (selectedItem != null)
        {
            selectedStoryListBox.Items.Remove(selectedItem);
            storyListBox.Items.Add(selectedItem);
            storyCountTextBox.Text = Convert.ToString(Convert.ToInt32(storyCountTextBox.Text) - 1);

            if (selectedStoryListBox.Items.Count == 0)
                removeStoryButton.Enabled = false;
        }
    }

    private void UpdateButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)teamListBox.SelectedItem;
        if (selectedItem != null)
        {
            currentStorylistBox.Items.Clear();
            storyCountTextBox.Enabled = true;
            storyListBox.Enabled = true;
            submitButton.Enabled = true;
            selectStoryButton.Enabled = true;
            removeStoryButton.Enabled = false;
            teamTextBox.Text = selectedItem;
            UpdateButton.Enabled = false;
        }
    }
  }
}
```

```
using System;
using System.IO;
using System.Threading;
using System.Windows.Forms;

namespace SDPTool
{
   class Team
   {
      static int TOTALMEMBERS = 2;
      static int TOTALSTORY = 50;
      int id;
      string name;
      Agent[] member = new Agent[TOTALMEMBERS];
      Thread[] memberThread = new Thread[TOTALMEMBERS];
      string[] storyNames = new string[TOTALSTORY];
      UserStory story;
      int storyCompleted = 0;
      int storyIndex = 0;
      int storyCount = 0;
      string activityType;
      bool startStory = true;
      double locAlloted = 0;
      Random rand = new Random();
      double b0 = 0;
      double b1 = 1;
      double estimatedLocPerHour = 0;
      Organization parent;

      string folderPath = "C:\\SDPTool\\Results\\";
      File summaryFile;
      double locPlanned = 0;
      double locDesigned = 0;
      double locDesignReviewed = 0;
      double locWritten = 0;
      double locCodeReviewed = 0;
      double locCompiled = 0;
      double locTested = 0;
      double locRefactored = 0;
      double locCompleted = 0;
      double currentDefects = 0;
      double totalDefectsRemoved = 0;
      double totalDefectsIntroduced = 0;

      public int Id
      {
         get  //get accessor method
         {
            return id;
         }
         set   //set accessor method
         {
```

```
        id = value;
    }
}

public string Name
{
    get  //get accessor method
    {
        return name;
    }
    set   //set accessor method
    {
        name = value;
    }
}

public UserStory Story
{
    get  //get accessor method
    {
        return story;
    }
    set   //set accessor method
    {
        story = value;
    }
}

public Organization Parent
{
    get  //get accessor method
    {
        return parent;
    }
    set   //set accessor method
    {
        parent = value;
    }
}

public string FolderPath
{
    get  //get accessor method
    {
        return folderPath;
    }
    set   //set accessor method
    {
        folderPath = value;
    }
}

public double B0
```

```
{
   get  //get accessor method
   {
      return b0;
   }
   set   //set accessor method
   {
      b0 = value;
   }
}

public double B1
{
   get  //get accessor method
   {
      return b1;
   }
   set   //set accessor method
   {
      b1 = value;
   }
}

public double EstimatedLocPerHour
{
   get  //get accessor method
   {
      return estimatedLocPerHour;
   }
   set   //set accessor method
   {
      estimatedLocPerHour = value;
   }
}

public double LocAlloted
{
   get  //get accessor method
   {
      return locAlloted;
   }
   set   //set accessor method
   {
      locAlloted = value;
   }
}

public double LocPlanned
{
   get  //get accessor method
   {
      return locPlanned;
   }
```

```
    set   //set accessor method
    {
        locPlanned = value;
    }
}

public double LocDesigned
{
    get  //get accessor method
    {
        return locDesigned;
    }
    set   //set accessor method
    {
        locDesigned = value;
    }
}

public double LocDesignReviewed
{
    get  //get accessor method
    {
        return locDesignReviewed;
    }
    set   //set accessor method
    {
        locDesignReviewed = value;
    }
}
public double LocWritten
{
    get  //get accessor method
    {
        return locWritten;
    }
    set   //set accessor method
    {
        locWritten = value;
    }
}

public double LocCodeReviewed
{
    get  //get accessor method
    {
        return locCodeReviewed;
    }
    set   //set accessor method
    {
        locCodeReviewed = value;
    }
}
public double LocCompiled
```

```
{
   get  //get accessor method
   {
      return locCompiled;
   }
   set   //set accessor method
   {
      locCompiled = value;
   }
}

public double LocTested
{
   get  //get accessor method
   {
      return locTested;
   }
   set   //set accessor method
   {
      locTested = value;
   }
}

public double LocRefactored
{
   get  //get accessor method
   {
      return locRefactored;
   }
   set   //set accessor method
   {
      locRefactored = value;
   }
}

public double LocCompleted
{
   get  //get accessor method
   {
      return locCompleted;
   }
   set   //set accessor method
   {
      locCompleted = value;
   }
}

public int StoryCompleted
{
   get  //get accessor method
   {
      return storyCompleted;
   }
```

```
    set   //set accessor method
    {
        storyCompleted = value;
    }
}

public double CurrentDefects
{
    get  //get accessor method
    {
        return currentDefects;
    }
    set   //set accessor method
    {
        currentDefects = value;
    }
}

public double TotalDefectsIntroduced
{
    get  //get accessor method
    {
        return totalDefectsIntroduced;
    }
    set   //set accessor method
    {
        totalDefectsIntroduced = value;
    }
}

public double TotalDefectsRemoved
{
    get  //get accessor method
    {
        return totalDefectsRemoved;
    }
    set   //set accessor method
    {
        totalDefectsRemoved = value;
    }
}

public string ActivityType
{
    get  //get accessor method
    {
        return activityType;
    }
    set   //set accessor method
    {
        activityType = value;
    }
}
```

```
public string[] StoryNames
{
   get  //get accessor method
   {
      return storyNames;
   }
   set   //set accessor method
   {
      storyNames = value;
   }
}

public int StoryCount
{
   get  //get accessor method
   {
      return storyCount;
   }
   set   //set accessor method
   {
      storyCount = value;
   }
}

public int StoryIndex
{
   get  //get accessor method
   {
      return storyIndex;
   }
   set   //set accessor method
   {
      storyIndex = value;
   }
}

public bool StartStory
{
   get  //get accessor method
   {
      return startStory;
   }
   set   //set accessor method
   {
      startStory = value;
   }
}

public void Run()
{
   //Run each agent...
   if ((startStory == true) || (story.Active.Equals(false)))
```

```
{
   if (storyIndex < storyCount)
   {
      story = parent.GetStory(name, storyNames[storyIndex]);
      while ((story.Active.Equals(false)) && (storyIndex < storyCount))
         story = parent.GetStory(name, storyNames[storyIndex++]);

      if (story.Active.Equals(true))
         activityType = story.ActivityType;
      else
         activityType = "Idle";

      startStory = false;
   }
}


if (story.Complete.Equals(false))
{
   story.B0 = B0;
   story.B1 = B1;
   story.Size = (B0 + story.EstimatedSize * B1) / 2;

   story.UpdateActivity();
   activityType = story.ActivityType;

   for (int i = 0; i < TOTALMEMBERS; i++)
   {
      member[i].ActivityType = activityType;
   }

   if (member[0].ActivityType.Equals("Design"))
   {
      member[rand.Next(2)].ActivityType = "DesignReview";
   }

   if (member[0].ActivityType.Equals("Code"))
   {
      member[rand.Next(2)].ActivityType = "CodeReview";
   }


   for (int i = 0; i < TOTALMEMBERS; i++)
   {
      //Create a new member thread and start running it
      memberThread[i] = new Thread(new ThreadStart(member[i].Run));
      memberThread[i].Start();
   }

   //block the current thread
   // Use the Join method to block the current thread
   // until the object's thread terminates.
   for (int i = 0; i < TOTALMEMBERS; i++)
```

```
        {
            memberThread[i].Join();
        }

        story.TimeSpent++;
        Summarize();
    }
    else
    {
        activityType = "Idle";
    }
}

public void Summarize()
{
    currentDefects = 0;
    totalDefectsIntroduced = 0;
    totalDefectsRemoved = 0;

    for (int i = 0; i < TOTALMEMBERS; i++)
    {
        //Calculations for the team
        locPlanned += member[i].LocPlanned;
        locDesigned += member[i].LocDesigned;
        locDesignReviewed += member[i].LocDesignReviewed;
        locWritten += member[i].LocWritten;
        locCodeReviewed += member[i].LocCodeReviewed;
        locCompiled += member[i].LocCompiled;
        locTested += member[i].LocTested;
        locRefactored += member[i].LocRefactored;
        locCompleted += member[i].LocCompleted;
        currentDefects += member[i].CurrentDefects;
        totalDefectsIntroduced += member[i].TotalDefectsIntroduced;
        totalDefectsRemoved += member[i].TotalDefectsRemoved;

        //Calculations for the UserStory
        story.LocPlanned += member[i].LocPlanned;
        story.LocDesigned += member[i].LocDesigned;
        story.LocDesignReviewed += member[i].LocDesignReviewed;
        story.LocWritten += member[i].LocWritten;
        story.LocCodeReviewed += member[i].LocCodeReviewed;
        story.LocCompiled += member[i].LocCompiled;
        story.LocTested += member[i].LocTested;
        story.LocRefactored += member[i].LocRefactored;
        story.LocCompleted += member[i].LocCompleted;
        story.CurrentDefects += member[i].DefectsIntroduced - member[i].DefectsRemoved;
    }

    if (locDesignReviewed > locDesigned)
    {
        locDesignReviewed = locDesigned;
        member[0].LocDesignReviewed = member[1].LocDesigned;
        member[1].LocDesignReviewed = member[0].LocDesigned;
```

```
            story.LocDesignReviewed = locDesigned;
        }

        if (locCodeReviewed > LocWritten)
        {
            locCodeReviewed = LocWritten;
            member[0].LocCodeReviewed = member[1].LocWritten;
            member[1].LocCodeReviewed = member[0].LocWritten;
            story.LocCodeReviewed = LocWritten;
        }

        if (story.CurrentDefects < 0)
            story.CurrentDefects = 0;

        if ((story.LocCompleted >= story.Size) && (story.CurrentDefects <= 0) &&
(story.Complete.Equals(false)))
        {
            story.Complete = true;
            story.ActivityType = "Completed";
            storyCompleted++;
            startStory = true;
            storyIndex++;
        }

        if (totalDefectsRemoved > totalDefectsIntroduced)
        {
            //check for each agent...
            if ((member[0].DefectsRemoved > 0) && (member[1].DefectsRemoved > 0))
            {
                member[0].TotalDefectsRemoved -= (totalDefectsRemoved - totalDefectsIntroduced) / 2;
                member[1].TotalDefectsRemoved -= (totalDefectsRemoved - totalDefectsIntroduced) / 2;
            }
            else if (member[0].DefectsRemoved > 0)
            {
                member[0].TotalDefectsRemoved -= (totalDefectsRemoved - totalDefectsIntroduced);
            }
            else if (member[1].DefectsRemoved > 0)
            {
                member[1].TotalDefectsRemoved -= (totalDefectsRemoved - totalDefectsIntroduced);
            }

            totalDefectsRemoved = totalDefectsIntroduced;
            currentDefects = 0;
        }

        if (currentDefects <= 0)
        {
            currentDefects = 0;
            for (int i = 0; i < TOTALMEMBERS; i++)
            {
                member[i].CurrentDefects = 0;
            }
        }
```

```
        }

        public Team(int ID, string Name, Agent agent1, Agent agent2, int sCount)
        {
            id = ID;
            name = Name;
            member[0] = agent1;
            member[1] = agent2;
            activityType = "Planning";
            locAlloted = 0;
            storyCount = sCount;

            for (int i = 0; i < TOTALMEMBERS; i++)
            {
                member[i].Parent = this;
                member[i].ActivityType = activityType;
            }

            string fileName = string.Concat(folderPath, name, "summary.txt");
            summaryFile = new File(fileName);
        }
    }
}
```

```
namespace SDPTool
{
   partial class StoryForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.label1 = new System.Windows.Forms.Label();
         this.TextBox1 = new System.Windows.Forms.TextBox();
         this.closeButton = new System.Windows.Forms.Button();
         this.clearAllButton = new System.Windows.Forms.Button();
         this.submitButton = new System.Windows.Forms.Button();
         this.addButton = new System.Windows.Forms.Button();
         this.storyListBox = new System.Windows.Forms.ListBox();
         this.removeButton = new System.Windows.Forms.Button();
         this.SuspendLayout();
         //
         // label1
         //
         this.label1.AutoSize = true;
         this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
         this.label1.Location = new System.Drawing.Point(12, 38);
         this.label1.Name = "label1";
         this.label1.Size = new System.Drawing.Size(158, 13);
         this.label1.TabIndex = 0;
         this.label1.Text = "Please enter a story name:";
         //
```

```
            // TextBox1
            //
            this.TextBox1.Location = new System.Drawing.Point(30, 71);
            this.TextBox1.Name = "TextBox1";
            this.TextBox1.Size = new System.Drawing.Size(125, 20);
            this.TextBox1.TabIndex = 1;
            this.TextBox1.TextChanged += new System.EventHandler(this.TextBox1_TextChanged);
            //
            // closeButton
            //
            this.closeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.closeButton.Location = new System.Drawing.Point(312, 185);
            this.closeButton.Name = "closeButton";
            this.closeButton.Size = new System.Drawing.Size(80, 30);
            this.closeButton.TabIndex = 6;
            this.closeButton.Text = "Close";
            this.closeButton.UseVisualStyleBackColor = true;
            this.closeButton.Click += new System.EventHandler(this.closeButton_Click);
            //
            // clearAllButton
            //
            this.clearAllButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.clearAllButton.Location = new System.Drawing.Point(217, 185);
            this.clearAllButton.Name = "clearAllButton";
            this.clearAllButton.Size = new System.Drawing.Size(80, 30);
            this.clearAllButton.TabIndex = 5;
            this.clearAllButton.Text = "Clear All";
            this.clearAllButton.UseVisualStyleBackColor = true;
            this.clearAllButton.Click += new System.EventHandler(this.clearAllButton_Click);
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(118, 185);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(80, 30);
            this.submitButton.TabIndex = 4;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // addButton
            //
            this.addButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.addButton.Location = new System.Drawing.Point(174, 65);
            this.addButton.Name = "addButton";
            this.addButton.Size = new System.Drawing.Size(54, 30);
            this.addButton.TabIndex = 2;
            this.addButton.Text = ">>";
```

```
            this.addButton.UseVisualStyleBackColor = true;
            this.addButton.Click += new System.EventHandler(this.addButton_Click);
            //
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(255, 38);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 108);
            this.storyListBox.TabIndex = 3;
            //
            // removeButton
            //
            this.removeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeButton.Location = new System.Drawing.Point(174, 101);
            this.removeButton.Name = "removeButton";
            this.removeButton.Size = new System.Drawing.Size(54, 30);
            this.removeButton.TabIndex = 7;
            this.removeButton.Text = "<<";
            this.removeButton.UseVisualStyleBackColor = true;
            this.removeButton.Click += new System.EventHandler(this.removeButton_Click);
            //
            // StoryForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(414, 247);
            this.Controls.Add(this.removeButton);
            this.Controls.Add(this.storyListBox);
            this.Controls.Add(this.addButton);
            this.Controls.Add(this.closeButton);
            this.Controls.Add(this.clearAllButton);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.TextBox1);
            this.Name = "StoryForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "StoryForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox TextBox1;
        private System.Windows.Forms.Button closeButton;
        private System.Windows.Forms.Button clearAllButton;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Button addButton;
        private System.Windows.Forms.ListBox storyListBox;
```

```
        private System.Windows.Forms.Button removeButton;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class StoryForm : Form
   {
      int storyCount;
      string folderPath = "C:\\SDPTool\\Results\\";
      string fileName;


      public string FolderPath
      {
         get  //get accessor method
         {
            return folderPath;
         }
         set   //set accessor method
         {
            folderPath = value;
         }
      }

      public StoryForm(string heading, int sCount)
      {
         InitializeComponent();
         if (heading.Equals("add"))
         {
            label1.Text = "Please add a story name:";
            addButton.Enabled = true;
            removeButton.Enabled = false;
         }
         else if (heading.Equals("remove"))
         {
            label1.Text = "Please remove a story name:";
            removeButton.Enabled = true;
            addButton.Enabled = false;
            TextBox1.Enabled = false;
         }
         else
         {
            label1.Text = "Please enter a story name:";
            addButton.Enabled = true;
            removeButton.Enabled = true;
         }
```

```
    fileName = folderPath + "userstory.txt";
    storyCount = sCount;
    TextBox1.Text = null;

    AddStoryList();
    if (storyListBox.Items.Count == storyCount)
    {
        submitButton.Enabled = true;

    }
    else
    {
        submitButton.Enabled = false;
    }
}

private void AddStoryList()
{
    FileStream inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
    StreamReader sr = new StreamReader(inFile);
    string s = sr.ReadLine();
    while (s != null)
    {
        storyListBox.Items.Add(s);
        s = sr.ReadLine();
    }
    sr.Close();
    inFile.Close();
}

private void submitButton_Click(object sender, EventArgs e)
{
    try
    {
        FileStream file;
        StreamWriter sw;
        file = new FileStream(fileName, FileMode.Create, FileAccess.Write);
        sw = new StreamWriter(file);

        foreach (object item in storyListBox.Items)
        {
            sw.WriteLine((string)item);
        }

        sw.Close();
        file.Close();
        this.Close();

    }
    catch
    {
```

```
            MessageBox.Show("The values are not valid. Please check the values again", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void addButton_Click(object sender, EventArgs e)
    {
        TextBox1.Text = TextBox1.Text.Trim();
        if (TextBox1.TextLength > 0)
        {
            storyListBox.Items.Add(TextBox1.Text);
            TextBox1.Text = null;
        }
        else
        {
            MessageBox.Show("Please enter a story name in the textbox");
        }

        if (storyListBox.Items.Count > 0)
            removeButton.Enabled = true;
        else
            removeButton.Enabled = false;

        if (storyListBox.Items.Count == storyCount)
        {
            submitButton.Enabled = true;
            addButton.Enabled = false;
        }
    }

    private void clearAllButton_Click(object sender, EventArgs e)
    {
        TextBox1.Clear();
        storyListBox.Text = null;
    }

    private void closeButton_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void TextBox1_TextChanged(object sender, EventArgs e)
    {
        if (TextBox1.Text.Trim().Length > 0)
            addButton.Enabled = true;
        else
            addButton.Enabled = false;
    }

    private void removeButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)storyListBox.SelectedItem;
        if (selectedItem != null)
```

```
        {
            storyListBox.Items.Remove(selectedItem);
            if (storyListBox.Items.Count == 0)
                removeButton.Enabled = false;
            if (storyListBox.Items.Count == storyCount)
            {
                submitButton.Enabled = true;
                addButton.Enabled = false;
            }
            else
            {
                submitButton.Enabled = false;
            }
        }
    }
}
```

```
namespace SDPTool
{
   partial class StoryEstimateForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.submitButton = new System.Windows.Forms.Button();
         this.label2 = new System.Windows.Forms.Label();
         this.textBox3 = new System.Windows.Forms.TextBox();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.label3 = new System.Windows.Forms.Label();
         this.label4 = new System.Windows.Forms.Label();
         this.textBox4 = new System.Windows.Forms.TextBox();
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.label1 = new System.Windows.Forms.Label();
         this.SuspendLayout();
         //
         // submitButton
         //
         this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
         this.submitButton.Location = new System.Drawing.Point(277, 147);
         this.submitButton.Name = "submitButton";
         this.submitButton.Size = new System.Drawing.Size(106, 30);
         this.submitButton.TabIndex = 4;
         this.submitButton.Text = "Submit";
         this.submitButton.UseVisualStyleBackColor = true;
```

```
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.Location = new System.Drawing.Point(24, 34);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(211, 13);
            this.label2.TabIndex = 8;
            this.label2.Text = "Please enter the estimates for story:";
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(207, 71);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(67, 20);
            this.textBox3.TabIndex = 0;
            this.textBox3.Text = "NA";
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(241, 31);
            this.textBox2.Name = "textBox2";
            this.textBox2.ReadOnly = true;
            this.textBox2.Size = new System.Drawing.Size(142, 20);
            this.textBox2.TabIndex = 7;
            this.textBox2.Text = "UserStory";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.Location = new System.Drawing.Point(24, 74);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(177, 13);
            this.label3.TabIndex = 1;
            this.label3.Text = "Estimated Story Size (in LOC):";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(24, 114);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(247, 13);
            this.label4.TabIndex = 3;
            this.label4.Text = "Estimated Story Completion Time (in mins):";
            //
            // textBox4
```

```
            //
            this.textBox4.Location = new System.Drawing.Point(277, 111);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(73, 20);
            this.textBox4.TabIndex = 2;
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(94, 6);
            this.textBox1.Name = "textBox1";
            this.textBox1.ReadOnly = true;
            this.textBox1.Size = new System.Drawing.Size(88, 20);
            this.textBox1.TabIndex = 5;
            this.textBox1.Text = "Team";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(24, 9);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(64, 13);
            this.label1.TabIndex = 6;
            this.label1.Text = "For Team:";
            //
            // StoryEstimateForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(399, 187);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.textBox3);
            this.Name = "StoryEstimateForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "StoryEstimateForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox3;
```

```
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label1;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SDPTool
{
   public partial class StoryEstimateForm : Form
   {
      double estimatedSize;
      double estimatedDuration;
      bool locKnown = false;

      public double EstimatedSize
      {
         get  //get accessor method
         {
            return estimatedSize;
         }
      }

      public double EstimatedDuration
      {
         get  //get accessor method
         {
            return estimatedDuration;
         }
      }

      public bool LocKnown
      {
         get  //get accessor method
         {
            return locKnown;
         }
      }

      public StoryEstimateForm(string teamName, string storyName)
      {
         InitializeComponent();
         textBox1.Text = teamName;
         textBox2.Text = storyName;
      }

      public StoryEstimateForm(string storyName)
      {
         InitializeComponent();
         label1.Visible = false;
         textBox1.Visible = false;
```

```
            textBox2.Text = storyName;
        }

        private void submitButton_Click(object sender, EventArgs e)
        {
            if ((textBox3.Text.Trim().Equals("NA")) || (textBox3.Text.Trim().Equals(null)))
            {
                estimatedSize = -1;
                locKnown = false;
            }
            else
            {
                estimatedSize = Convert.ToDouble(textBox3.Text);
                locKnown = true;
            }
            estimatedDuration = Convert.ToDouble(textBox4.Text);
            Close();
        }
    }
}
```

```
namespace SDPTool
{
   partial class SimulationForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.stopButton = new System.Windows.Forms.Button();
         this.goAgentButton = new System.Windows.Forms.Button();
         this.exitButton = new System.Windows.Forms.Button();
         this.menuStrip1 = new System.Windows.Forms.MenuStrip();
         this.fileToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.interfaceMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.windowsMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.consoleMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.exitToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.editToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.createToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.organizationMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.agentMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.teamsMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.simulationToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.initializeToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.startToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.stopToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.helpToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.aboutToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
         this.label2 = new System.Windows.Forms.Label();
```

```
            this.label4 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.statusTextBox = new System.Windows.Forms.RichTextBox();
            this.expectedTextBox = new System.Windows.Forms.RichTextBox();
            this.actualTextBox = new System.Windows.Forms.RichTextBox();
            this.yesButton = new System.Windows.Forms.Button();
            this.queryTextBox = new System.Windows.Forms.RichTextBox();
            this.noButton = new System.Windows.Forms.Button();
            this.agentTextBox = new System.Windows.Forms.RichTextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.agentListBox = new System.Windows.Forms.ListBox();
            this.teamTextBox = new System.Windows.Forms.RichTextBox();
            this.organizationTextBox = new System.Windows.Forms.RichTextBox();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.teamListBox = new System.Windows.Forms.ListBox();
            this.label9 = new System.Windows.Forms.Label();
            this.goTeamButton = new System.Windows.Forms.Button();
            this.storyListBox = new System.Windows.Forms.ListBox();
            this.label10 = new System.Windows.Forms.Label();
            this.goStoryButton = new System.Windows.Forms.Button();
            this.storyTextBox = new System.Windows.Forms.RichTextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.startButton = new System.Windows.Forms.Button();
            this.aboutButton = new System.Windows.Forms.Button();
            this.menuStrip1.SuspendLayout();
            this.SuspendLayout();
            //
            // stopButton
            //
            this.stopButton.Location = new System.Drawing.Point(823, 569);
            this.stopButton.Name = "stopButton";
            this.stopButton.Size = new System.Drawing.Size(75, 23);
            this.stopButton.TabIndex = 28;
            this.stopButton.Text = "Stop";
            this.stopButton.UseVisualStyleBackColor = true;
            this.stopButton.Visible = false;
            this.stopButton.Click += new System.EventHandler(this.stopButton_Click);
            //
            // goAgentButton
            //
            this.goAgentButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.goAgentButton.ForeColor = System.Drawing.Color.Green;
            this.goAgentButton.Location = new System.Drawing.Point(933, 516);
            this.goAgentButton.Name = "goAgentButton";
            this.goAgentButton.Size = new System.Drawing.Size(42, 43);
            this.goAgentButton.TabIndex = 27;
            this.goAgentButton.Text = "Go";
            this.goAgentButton.UseVisualStyleBackColor = true;
            this.goAgentButton.Click += new System.EventHandler(this.goAgentButton_Click);
```

```
            //
            // exitButton
            //
            this.exitButton.Location = new System.Drawing.Point(904, 569);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(75, 23);
            this.exitButton.TabIndex = 29;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // menuStrip1
            //
            this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.fileToolStripMenuItem,
            this.editToolStripMenuItem,
            this.simulationToolStripMenuItem,
            this.helpToolStripMenuItem});
            this.menuStrip1.Location = new System.Drawing.Point(0, 0);
            this.menuStrip1.Name = "menuStrip1";
            this.menuStrip1.Size = new System.Drawing.Size(992, 24);
            this.menuStrip1.TabIndex = 0;
            this.menuStrip1.Text = "menuStrip1";
            this.menuStrip1.Visible = false;
            //
            // fileToolStripMenuItem
            //
            this.fileToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.interfaceMenuItem,
            this.exitToolStripMenuItem});
            this.fileToolStripMenuItem.Name = "fileToolStripMenuItem";
            this.fileToolStripMenuItem.Size = new System.Drawing.Size(35, 20);
            this.fileToolStripMenuItem.Text = "File";
            //
            // interfaceMenuItem
            //
            this.interfaceMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[]
{
            this.windowsMenuItem,
            this.consoleMenuItem});
            this.interfaceMenuItem.Name = "interfaceMenuItem";
            this.interfaceMenuItem.Size = new System.Drawing.Size(119, 22);
            this.interfaceMenuItem.Text = "Interface";
            //
            // windowsMenuItem
            //
            this.windowsMenuItem.Name = "windowsMenuItem";
            this.windowsMenuItem.Size = new System.Drawing.Size(117, 22);
            this.windowsMenuItem.Text = "Windows";
            this.windowsMenuItem.Click += new System.EventHandler(this.windowsMenuItem_Click);
            //
            // consoleMenuItem
```

```
            //
            this.consoleMenuItem.Name = "consoleMenuItem";
            this.consoleMenuItem.Size = new System.Drawing.Size(117, 22);
            this.consoleMenuItem.Text = "Console";
            //
            // exitToolStripMenuItem
            //
            this.exitToolStripMenuItem.Name = "exitToolStripMenuItem";
            this.exitToolStripMenuItem.Size = new System.Drawing.Size(119, 22);
            this.exitToolStripMenuItem.Text = "Exit";
            this.exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
            //
            // editToolStripMenuItem
            //
            this.editToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.createToolStripMenuItem});
            this.editToolStripMenuItem.Name = "editToolStripMenuItem";
            this.editToolStripMenuItem.Size = new System.Drawing.Size(37, 20);
            this.editToolStripMenuItem.Text = "Edit";
            //
            // createToolStripMenuItem
            //
            this.createToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.organizationMenuItem,
            this.agentMenuItem,
            this.teamsMenuItem});
            this.createToolStripMenuItem.Name = "createToolStripMenuItem";
            this.createToolStripMenuItem.Size = new System.Drawing.Size(107, 22);
            this.createToolStripMenuItem.Text = "Create";
            //
            // organizationMenuItem
            //
            this.organizationMenuItem.Name = "organizationMenuItem";
            this.organizationMenuItem.Size = new System.Drawing.Size(135, 22);
            this.organizationMenuItem.Text = "Organization";
            this.organizationMenuItem.Click += new System.EventHandler(this.organizationMenuItem_Click);
            //
            // agentMenuItem
            //
            this.agentMenuItem.Name = "agentMenuItem";
            this.agentMenuItem.Size = new System.Drawing.Size(135, 22);
            this.agentMenuItem.Text = "Agent";
            this.agentMenuItem.Click += new System.EventHandler(this.agentMenuItem_Click);
            //
            // teamsMenuItem
            //
            this.teamsMenuItem.Name = "teamsMenuItem";
            this.teamsMenuItem.Size = new System.Drawing.Size(135, 22);
            this.teamsMenuItem.Text = "Teams";
            this.teamsMenuItem.Click += new System.EventHandler(this.teamsMenuItem_Click);
```

```
            //
            // simulationToolStripMenuItem
            //
            this.simulationToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.initializeToolStripMenuItem,
            this.startToolStripMenuItem,
            this.stopToolStripMenuItem});
            this.simulationToolStripMenuItem.Name = "simulationToolStripMenuItem";
            this.simulationToolStripMenuItem.Size = new System.Drawing.Size(67, 20);
            this.simulationToolStripMenuItem.Text = "Simulation";
            //
            // initializeToolStripMenuItem
            //
            this.initializeToolStripMenuItem.Name = "initializeToolStripMenuItem";
            this.initializeToolStripMenuItem.Size = new System.Drawing.Size(113, 22);
            this.initializeToolStripMenuItem.Text = "Initialize";
            this.initializeToolStripMenuItem.Click += new
System.EventHandler(this.initializeToolStripMenuItem_Click);
            //
            // startToolStripMenuItem
            //
            this.startToolStripMenuItem.Name = "startToolStripMenuItem";
            this.startToolStripMenuItem.Size = new System.Drawing.Size(113, 22);
            this.startToolStripMenuItem.Text = "Start";
            this.startToolStripMenuItem.Click += new
System.EventHandler(this.startToolStripMenuItem_Click);
            //
            // stopToolStripMenuItem
            //
            this.stopToolStripMenuItem.Name = "stopToolStripMenuItem";
            this.stopToolStripMenuItem.Size = new System.Drawing.Size(113, 22);
            this.stopToolStripMenuItem.Text = "Stop";
            this.stopToolStripMenuItem.Click += new
System.EventHandler(this.stopToolStripMenuItem_Click);
            //
            // helpToolStripMenuItem
            //
            this.helpToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.aboutToolStripMenuItem});
            this.helpToolStripMenuItem.Name = "helpToolStripMenuItem";
            this.helpToolStripMenuItem.Size = new System.Drawing.Size(40, 20);
            this.helpToolStripMenuItem.Text = "Help";
            //
            // aboutToolStripMenuItem
            //
            this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
            this.aboutToolStripMenuItem.Size = new System.Drawing.Size(103, 22);
            this.aboutToolStripMenuItem.Text = "About";
            this.aboutToolStripMenuItem.Click += new
System.EventHandler(this.aboutToolStripMenuItem_Click);
            //
```

```
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.Location = new System.Drawing.Point(672, 35);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(90, 13);
            this.label2.TabIndex = 9;
            this.label2.Text = "Expected Values:";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(672, 164);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(126, 13);
            this.label4.TabIndex = 11;
            this.label4.Text = "Actual Simulation Values:";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(12, 35);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(91, 13);
            this.label1.TabIndex = 1;
            this.label1.Text = "Simulation Status:";
            //
            // statusTextBox
            //
            this.statusTextBox.Location = new System.Drawing.Point(15, 51);
            this.statusTextBox.Name = "statusTextBox";
            this.statusTextBox.ReadOnly = true;
            this.statusTextBox.Size = new System.Drawing.Size(300, 180);
            this.statusTextBox.TabIndex = 2;
            this.statusTextBox.Text = "";
            //
            // expectedTextBox
            //
            this.expectedTextBox.Location = new System.Drawing.Point(675, 51);
            this.expectedTextBox.Name = "expectedTextBox";
            this.expectedTextBox.ReadOnly = true;
            this.expectedTextBox.Size = new System.Drawing.Size(300, 110);
            this.expectedTextBox.TabIndex = 10;
            this.expectedTextBox.Text = "";
            //
            // actualTextBox
            //
```

```
            this.actualTextBox.Location = new System.Drawing.Point(675, 180);
            this.actualTextBox.Name = "actualTextBox";
            this.actualTextBox.ReadOnly = true;
            this.actualTextBox.Size = new System.Drawing.Size(300, 110);
            this.actualTextBox.TabIndex = 12;
            this.actualTextBox.Text = "";
            //
            // yesButton
            //
            this.yesButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.yesButton.ForeColor = System.Drawing.Color.Green;
            this.yesButton.Location = new System.Drawing.Point(279, 246);
            this.yesButton.Name = "yesButton";
            this.yesButton.Size = new System.Drawing.Size(36, 20);
            this.yesButton.TabIndex = 5;
            this.yesButton.Text = "Yes";
            this.yesButton.UseVisualStyleBackColor = true;
            this.yesButton.Click += new System.EventHandler(this.yesButton_Click);
            //
            // queryTextBox
            //
            this.queryTextBox.BackColor = System.Drawing.SystemColors.Info;
            this.queryTextBox.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.queryTextBox.ForeColor = System.Drawing.Color.Red;
            this.queryTextBox.Location = new System.Drawing.Point(15, 247);
            this.queryTextBox.Name = "queryTextBox";
            this.queryTextBox.ReadOnly = true;
            this.queryTextBox.Size = new System.Drawing.Size(264, 43);
            this.queryTextBox.TabIndex = 4;
            this.queryTextBox.Text = "";
            //
            // noButton
            //
            this.noButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.noButton.ForeColor = System.Drawing.Color.Red;
            this.noButton.Location = new System.Drawing.Point(279, 270);
            this.noButton.Name = "noButton";
            this.noButton.Size = new System.Drawing.Size(36, 20);
            this.noButton.TabIndex = 6;
            this.noButton.Text = "No";
            this.noButton.UseVisualStyleBackColor = true;
            this.noButton.Click += new System.EventHandler(this.noButton_Click);
            //
            // agentTextBox
            //
            this.agentTextBox.Location = new System.Drawing.Point(675, 314);
            this.agentTextBox.Name = "agentTextBox";
            this.agentTextBox.ReadOnly = true;
            this.agentTextBox.Size = new System.Drawing.Size(300, 180);
            this.agentTextBox.TabIndex = 24;
```

```
            this.agentTextBox.Text = "";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.Location = new System.Drawing.Point(672, 298);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(101, 13);
            this.label3.TabIndex = 23;
            this.label3.Text = "Agent Performance:";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label6.Location = new System.Drawing.Point(12, 231);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(40, 13);
            this.label6.TabIndex = 3;
            this.label6.Text = "Dialog:";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label5.Location = new System.Drawing.Point(672, 500);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(57, 13);
            this.label5.TabIndex = 25;
            this.label5.Text = "Agent List:";
            //
            // agentListBox
            //
            this.agentListBox.BackColor = System.Drawing.SystemColors.Info;
            this.agentListBox.FormattingEnabled = true;
            this.agentListBox.Location = new System.Drawing.Point(675, 516);
            this.agentListBox.Name = "agentListBox";
            this.agentListBox.Size = new System.Drawing.Size(258, 43);
            this.agentListBox.TabIndex = 26;
            //
            // teamTextBox
            //
            this.teamTextBox.Location = new System.Drawing.Point(345, 314);
            this.teamTextBox.Name = "teamTextBox";
            this.teamTextBox.ReadOnly = true;
            this.teamTextBox.Size = new System.Drawing.Size(300, 180);
            this.teamTextBox.TabIndex = 19;
            this.teamTextBox.Text = "";
            //
```

```
            // organizationTextBox
            //
            this.organizationTextBox.Location = new System.Drawing.Point(345, 51);
            this.organizationTextBox.Name = "organizationTextBox";
            this.organizationTextBox.ReadOnly = true;
            this.organizationTextBox.Size = new System.Drawing.Size(300, 239);
            this.organizationTextBox.TabIndex = 8;
            this.organizationTextBox.Text = "";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(342, 298);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(100, 13);
            this.label7.TabIndex = 18;
            this.label7.Text = "Team Performance:";
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label8.Location = new System.Drawing.Point(342, 35);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(132, 13);
            this.label8.TabIndex = 7;
            this.label8.Text = "Organization Performance:";
            //
            // teamListBox
            //
            this.teamListBox.BackColor = System.Drawing.SystemColors.Info;
            this.teamListBox.FormattingEnabled = true;
            this.teamListBox.Location = new System.Drawing.Point(345, 516);
            this.teamListBox.Name = "teamListBox";
            this.teamListBox.Size = new System.Drawing.Size(258, 43);
            this.teamListBox.TabIndex = 21;
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label9.Location = new System.Drawing.Point(342, 500);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(56, 13);
            this.label9.TabIndex = 20;
            this.label9.Text = "Team List:";
            //
            // goTeamButton
            //
```

```
            this.goTeamButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.goTeamButton.ForeColor = System.Drawing.Color.Green;
            this.goTeamButton.Location = new System.Drawing.Point(603, 516);
            this.goTeamButton.Name = "goTeamButton";
            this.goTeamButton.Size = new System.Drawing.Size(42, 43);
            this.goTeamButton.TabIndex = 22;
            this.goTeamButton.Text = "Go";
            this.goTeamButton.UseVisualStyleBackColor = true;
            this.goTeamButton.Click += new System.EventHandler(this.goTeamButton_Click);
            //
            // storyListBox
            //
            this.storyListBox.BackColor = System.Drawing.SystemColors.Info;
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(15, 516);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(258, 43);
            this.storyListBox.TabIndex = 16;
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label10.Location = new System.Drawing.Point(12, 500);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(53, 13);
            this.label10.TabIndex = 15;
            this.label10.Text = "Story List:";
            //
            // goStoryButton
            //
            this.goStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.goStoryButton.ForeColor = System.Drawing.Color.Green;
            this.goStoryButton.Location = new System.Drawing.Point(273, 516);
            this.goStoryButton.Name = "goStoryButton";
            this.goStoryButton.Size = new System.Drawing.Size(42, 43);
            this.goStoryButton.TabIndex = 17;
            this.goStoryButton.Text = "Go";
            this.goStoryButton.UseVisualStyleBackColor = true;
            this.goStoryButton.Click += new System.EventHandler(this.goStoryButton_Click);
            //
            // storyTextBox
            //
            this.storyTextBox.Location = new System.Drawing.Point(15, 314);
            this.storyTextBox.Name = "storyTextBox";
            this.storyTextBox.ReadOnly = true;
            this.storyTextBox.Size = new System.Drawing.Size(300, 180);
            this.storyTextBox.TabIndex = 14;
            this.storyTextBox.Text = "";
            //
```

```
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label11.Location = new System.Drawing.Point(12, 298);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(86, 13);
            this.label11.TabIndex = 13;
            this.label11.Text = "User Story Stats:";
            //
            // startButton
            //
            this.startButton.Location = new System.Drawing.Point(823, 569);
            this.startButton.Name = "startButton";
            this.startButton.Size = new System.Drawing.Size(75, 23);
            this.startButton.TabIndex = 30;
            this.startButton.Text = "Start";
            this.startButton.UseVisualStyleBackColor = true;
            this.startButton.Click += new System.EventHandler(this.startButton_Click);
            //
            // aboutButton
            //
            this.aboutButton.Location = new System.Drawing.Point(742, 569);
            this.aboutButton.Name = "aboutButton";
            this.aboutButton.Size = new System.Drawing.Size(75, 23);
            this.aboutButton.TabIndex = 31;
            this.aboutButton.Text = "About";
            this.aboutButton.UseVisualStyleBackColor = true;
            this.aboutButton.Click += new System.EventHandler(this.aboutButton_Click);
            //
            // SimulationForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.AutoScroll = true;
            this.AutoSize = true;
            this.ClientSize = new System.Drawing.Size(992, 601);
            this.Controls.Add(this.aboutButton);
            this.Controls.Add(this.startButton);
            this.Controls.Add(this.storyListBox);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.goStoryButton);
            this.Controls.Add(this.storyTextBox);
            this.Controls.Add(this.label11);
            this.Controls.Add(this.teamListBox);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.goTeamButton);
            this.Controls.Add(this.teamTextBox);
            this.Controls.Add(this.organizationTextBox);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.agentListBox);
```

```
    this.Controls.Add(this.label5);
    this.Controls.Add(this.label6);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.agentTextBox);
    this.Controls.Add(this.noButton);
    this.Controls.Add(this.queryTextBox);
    this.Controls.Add(this.yesButton);
    this.Controls.Add(this.actualTextBox);
    this.Controls.Add(this.expectedTextBox);
    this.Controls.Add(this.statusTextBox);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.exitButton);
    this.Controls.Add(this.goAgentButton);
    this.Controls.Add(this.stopButton);
    this.Controls.Add(this.menuStrip1);
    this.MainMenuStrip = this.menuStrip1;
    this.Name = "SimulationForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
    this.Text = "Software Development Process Simulation";
    this.menuStrip1.ResumeLayout(false);
    this.menuStrip1.PerformLayout();
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button stopButton;
private System.Windows.Forms.Button goAgentButton;
private System.Windows.Forms.Button exitButton;
private System.Windows.Forms.MenuStrip menuStrip1;
private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem interfaceMenuItem;
private System.Windows.Forms.ToolStripMenuItem editToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem createToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem organizationMenuItem;
private System.Windows.Forms.ToolStripMenuItem agentMenuItem;
private System.Windows.Forms.ToolStripMenuItem simulationToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem startToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem stopToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem helpToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem aboutToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem windowsMenuItem;
private System.Windows.Forms.ToolStripMenuItem consoleMenuItem;
private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem teamsMenuItem;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.ToolStripMenuItem initializeToolStripMenuItem;
```

```
        private System.Windows.Forms.RichTextBox statusTextBox;
        private System.Windows.Forms.RichTextBox expectedTextBox;
        private System.Windows.Forms.RichTextBox actualTextBox;
        private System.Windows.Forms.Button yesButton;
        private System.Windows.Forms.RichTextBox queryTextBox;
        private System.Windows.Forms.Button noButton;
        private System.Windows.Forms.RichTextBox agentTextBox;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.ListBox agentListBox;
        private System.Windows.Forms.RichTextBox teamTextBox;
        private System.Windows.Forms.RichTextBox organizationTextBox;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.ListBox teamListBox;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Button goTeamButton;
        private System.Windows.Forms.ListBox storyListBox;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Button goStoryButton;
        private System.Windows.Forms.RichTextBox storyTextBox;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Button startButton;
        private System.Windows.Forms.Button aboutButton;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class SimulationForm : Form
   {
      static int TOTALPARAMETERS = 7;
      static int TOTALSTORY = 50;
      static int TOTALMEASURES = 10;
      static int DEFAULT_INTERVAL = 1; //1 min in Simulation time
      bool pauseEnabled = false;
      bool stop = false;


      string folderPath;
      File summaryFile;
      char[] delimeter = new char[] { ',' };
      string agentFile;
      string teamFile;
      string initializeFile;
      Organization org;
      int agentCount;
      int teamCount;
      int storyCount;
      string[] storyName = new string[TOTALSTORY];
      double interval = 0;
      double projectSize;
      string[] measurementList = new string[TOTALMEASURES];
      int measurementCount = 0;
      bool IAmPaused = true;
      double time = 0;
      double intervalCount = 0;
      InitializeForm initializeForm1;
      StoryForm storyForm1;
      AgentForm agentForm1;
      TeamForm teamForm1;
      MeasurementForm measurementForm1;
      UpdateTeamForm teamUpdateForm1;
      UpdateForm updateForm1;

      public SimulationForm()
      {
         EnsureDirectory(new System.IO.DirectoryInfo("C:\\SDPTool\\Results"));
         folderPath = "C:\\SDPTool\\Results\\";
         summaryFile = new File(folderPath + "summary.txt");
```

```
            teamFile = folderPath + "teams.txt";
            initializeFile = folderPath + "initialize.txt";
            InitializeComponent();
            DisableAllButtons();
        }

        public static void EnsureDirectory(System.IO.DirectoryInfo oDirInfo)
        {
            if (oDirInfo.Parent != null)
                EnsureDirectory(oDirInfo.Parent);
            if (!oDirInfo.Exists)
            {
                oDirInfo.Create();
            }
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ExitSimulation();
        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            ExitSimulation();
        }

        private void ExitSimulation()
        {
            DialogResult result = MessageBox.Show("Are you sure you want to Exit the Simulation", "Alert",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (result.Equals(DialogResult.Yes))
            {
                this.Close();
                Environment.Exit(0);
            }
        }

        private void stopToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StopSimulation();
        }

        private void stopButton_Click(object sender, EventArgs e)
        {
            StopSimulation();
            startButton.Visible = true;
            stopButton.Visible = false;
        }

        private void StopSimulation()
        {
            DialogResult result = MessageBox.Show("Are you sure you want to Stop the Simulation", "Alert",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
```

```
    if (result.Equals(DialogResult.Yes))
    {
        //Stop the Simulation
        stop = true;
        yesButton.Enabled = false;
        noButton.Enabled = false;
        queryTextBox.Text = null;
        queryTextBox.Enabled = false;
    }
}

private void windowsMenuItem_Click(object sender, EventArgs e)
{
    agentForm1 = new AgentForm();
    agentForm1.ShowDialog();
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    AboutForm aboutForm1 = new AboutForm();
    aboutForm1.Show();
}

private void agentMenuItem_Click(object sender, EventArgs e)
{
    agentForm1 = new AgentForm();
    agentForm1.ShowDialog();

    agentFile = folderPath + "Employees.txt";
}

private void teamsMenuItem_Click(object sender, EventArgs e)
{
    teamForm1 = new TeamForm(teamCount, storyCount);

    teamForm1.ShowDialog();
}

private void initializeToolStripMenuItem_Click(object sender, EventArgs e)
{
    InitializeSimulation();
}

private void AddStoryList()
{
    int sCount = 0;
    int i = -1;
    storyName.Initialize();
    string fileName = folderPath + "userstory.txt";
    FileStream inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
    StreamReader sr = new StreamReader(inFile);
    string s = sr.ReadLine();
    while ((s != null) && (sCount < storyCount))
```

```
    {
        storyName[++i] = s;
        storyListBox.Items.Add(s);
        teamForm1.AddStory(s);
        s = sr.ReadLine();
    }
    sr.Close();
    inFile.Close();
}

private void AddAgentList()
{
    agentListBox.Items.Clear();
    agentTextBox.Text = null;
    for (int i = 0; i < org.AgentCount; i++)
    {
        agentListBox.Items.Add(org.Agent[i].Name);
        teamForm1.AddAgent(org.Agent[i].Name);
    }
}

private void AddTeamList()
{
    teamListBox.Items.Clear();
    teamTextBox.Text = null;
    for (int i = 0; i < org.TeamCount; i++)
    {
        teamListBox.Items.Add(org.Team[i].Name);
    }
}

private void InitializeSimulation()
{
    //Initialize the Simulation Components
    initializeForm1 = new InitializeForm();
    initializeForm1.FileName = initializeFile;
    initializeForm1.ShowDialog();

    string[] parameters = new string[TOTALPARAMETERS + 1];
    string[] words = new string[TOTALPARAMETERS];
    FileStream inFile = null;
    StreamReader sr = null;
    string line = null;

    try
    {
        inFile = new FileStream(initializeFile, FileMode.OpenOrCreate, FileAccess.Read);
        sr = new StreamReader(inFile);
        line = sr.ReadLine();
        words = line.Split(delimeter);

        int i = 0;
        foreach (string word in words)
```

```
            {
                parameters[++i] = word;
            }

            if (i == TOTALPARAMETERS)
            {
                agentCount = Int32.Parse(parameters[1]);
                teamCount = Int32.Parse(parameters[2]);
                if (Double.Parse(parameters[3]).Equals(-1))
                {
                    //LOC not known
                    projectSize = -1;
                }
                else
                {
                    projectSize = Double.Parse(parameters[3]);
                }
                storyCount = Int32.Parse(parameters[4]);
                pauseEnabled = bool.Parse(parameters[5]);
                interval = Double.Parse(parameters[6]);
                if (bool.Parse(parameters[7]))
                    agentFile = "Agents.txt";
                else
                    agentFile = folderPath + "Employees.txt";

                if (teamCount > agentCount / 2)
                {
                    if (MessageBox.Show("Do you want to continue with \nTeam Count = Employee Pairs ???",
"Error: Team Count > Employee pairs.", MessageBoxButtons.YesNo, MessageBoxIcon.Error) ==
DialogResult.Yes)
                        teamCount = agentCount / 2;
                    else
                        ExitSimulation();
                }

            }
            else
            {
                MessageBox.Show("initialize.txt is not valid. Please re-check the values", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                inFile = null;
            }
            sr.Close();
            inFile.Close();
        }
        catch (Exception ex)
        {
            if (inFile != null)
            {
                inFile.Close();
                inFile = null;
            }
```

```
            MessageBox.Show("initialize.txt is not valid. Please re-check the values", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            Console.WriteLine(ex.ToString());
        }
        finally
        {
            if (inFile != null)
            {
                inFile.Close();
                GetMeasurementsList();
                storyForm1 = new StoryForm("new", storyCount);
                storyForm1.FolderPath = folderPath;
                storyForm1.ShowDialog();
            }
            else
            {
                initializeForm1.Dispose();
                InitializeSimulation();
            }
        }
    }

    private void organizationMenuItem_Click(object sender, EventArgs e)
    {
        CreateOrganization();
    }

    private void CreateOrganization()
    {
        InitializeSimulation();
        org = new Organization(agentFile, delimeter, agentCount);
        org.FolderPath = folderPath;
        summaryFile.WriteLine(string.Concat(Convert.ToString(org.AgentCount), " Agents", " Created"));

        teamForm1 = new TeamForm(teamCount, storyCount);
        teamForm1.FolderPath = folderPath;

        AddAgentList();
        AddStoryList();
        org.CreateStories(storyName);
        if (teamCount > 0)
        {
            teamForm1.ShowDialog();
            org.CreateTeams(teamFile, delimeter, teamCount);
        }
        else
        {
            teamForm1.ShowDialog();
        }
        AddTeamList();
    }

    private void GetMeasurementsList()
```

```
{
    measurementForm1 = new MeasurementForm();
    measurementForm1.ShowDialog();
    measurementList = measurementForm1.Measures;
    measurementCount = measurementForm1.Count;
}

private void startToolStripMenuItem_Click(object sender, EventArgs e)
{
    StartSimulation();
}


private void startButton_Click(object sender, EventArgs e)
{
    StartSimulation();
}

private void StartSimulation()
{
    //Start the Simulation
    startButton.Visible = false;
    stopButton.Visible = true;
    startToolStripMenuItem.Enabled = false;
    initializeToolStripMenuItem.Enabled = false;

    // Initialize everything for every new start
    stop = false;
    agentCount = 0;
    teamCount = 0;
    interval = 0;
    projectSize = 0;
    IAmPaused = true;
    time = 0;
    queryTextBox.Clear();
    CreateOrganization();
    stopButton.Visible = true;
    int i = 0;

    if (interval == 0)
        interval = DEFAULT_INTERVAL;

    org.ProjectSize = projectSize;

    statusTextBox.Text = "Simulation Started.\n";

    while ((!stop) && ((org.StoryCompleted < storyCount) || (org.CurrentDefects > 0)))
    {
        statusTextBox.AppendText("\nInterval Started. ");

        i = 1;
        intervalCount++;
```

```
            while ((!stop) && ((i <= interval) && ((org.StoryCompleted < storyCount) || (org.CurrentDefects
> 0))))
            {
               Application.DoEvents();
               time++;
               org.Run();
               DisplayStatus();
               i = i + 1;
            }

            statusTextBox.AppendText("Interval End. \n");

            //Summarize the results after the interval
            org.Summarize();
            DisplayStatus();
            DisplayOrganizationStatus();
            DisplayExpectedValues();
            DisplayActualValues();

            //To enable the sumulation to run with/without pausing...
            if ((!stop) && pauseEnabled)
            {
               if ((org.StoryCompleted < storyCount) || (org.CurrentDefects > 0))
               {
                  queryTextBox.Text = "Want to make any changes to the Simulation?";
                  statusTextBox.AppendText("\nWaiting for your response...\n");
                  statusTextBox.ScrollToCaret();
                  EnableAllButtons();
                  IAmPaused = true;
                  while ((!stop) && IAmPaused)
                  {
                     Application.DoEvents();
                  }
               }
            }
         }

         org.Summarize();

         //Simulation Ends!!!

         if ((org.StoryCompleted >= storyCount) && (i != 0))
         {
            //Final Status Displayed...
            DisplayStatus();
            DisplayOrganizationStatus();
            DisplayExpectedValues();
            DisplayActualValues();
            MessageBox.Show("Simulation Over. Thank you!!!", "Simulation Over!!!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            statusTextBox.AppendText("\n\nSimulation End.\n");
         }
         else
```

```
        {
            MessageBox.Show("Simulation Stopped", "Aborted", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            statusTextBox.AppendText("\n\nSimulation Stopped.\n");
        }

        statusTextBox.ScrollToCaret();
        goStoryButton.Enabled = true;
        goAgentButton.Enabled = true;
        goTeamButton.Enabled = true;
        startToolStripMenuItem.Enabled = true;
        initializeToolStripMenuItem.Enabled = true;
        stopButton.Visible = false;
    }

    private void DisplayStatus()
    {
        if (time >= 60)
        {
            statusTextBox.AppendText(string.Concat("\nTime: ", Convert.ToString((int)(time / 60)), " hours
and ", Convert.ToString(time - 60 * (int)(time / 60)), " mins. \n"));
            summaryFile.WriteLine(string.Concat("\nTime: ", Convert.ToString((int)(time / 60)), " hours and
", Convert.ToString(time - 60 * (int)(time / 60)), " mins. \n"));
        }
        else
        {
            statusTextBox.AppendText(string.Concat("\nTime: ", Convert.ToString(time), " mins. \n"));
            summaryFile.WriteLine(string.Concat("\nTime: ", Convert.ToString(time), " mins. \n"));
        }
        summaryFile.WriteLine(string.Concat("Total of ", Convert.ToString(org.LocWritten), " LOC
written\n"));
        summaryFile.WriteLine(string.Concat("LOC Completed: ", Convert.ToString(org.LocCompleted),
".\n"));
        summaryFile.WriteLine(string.Concat("Stories Completed: ",
Convert.ToString(org.StoryCompleted), ".\n"));
        summaryFile.WriteLine("Story Details: \n");
        foreach (string item in storyName)
        {
            if (item != null)
            {
                UserStory story = org.GetStory(item);
                if ((story.Active.Equals(true)) && (story.Complete.Equals(true)) && (story.EstimatedSize >
0))
                {
                    summaryFile.WriteLine(string.Concat("Story: ", story.Name, "Duration: ", story.TimeSpent,
"mins. \n"));
                }
            }
        }
        summaryFile.WriteLine(string.Concat("Total of ", Convert.ToString(org.CurrentDefects), " defects
present.\n"));
        summaryFile.WriteLine(string.Concat("Total of ", Convert.ToString(org.TotalDefectsIntroduced), "
total defects introduced.\n"));
```

```
            summaryFile.WriteLine(string.Concat("Total of ", Convert.ToString(org.TotalDefectsRemoved), "
total defects removed.\n"));

            statusTextBox.AppendText(string.Concat("Total of ", Convert.ToString(org.LocWritten), " LOC
written\n"));
            statusTextBox.AppendText(string.Concat("LOC Completed: ",
Convert.ToString(org.LocCompleted), ".\n"));
            statusTextBox.AppendText(string.Concat("Stories Completed: ",
Convert.ToString(org.StoryCompleted), ".\n"));
            statusTextBox.AppendText(string.Concat("Total of ", Convert.ToString(org.CurrentDefects), "
defects present.\n"));
            statusTextBox.AppendText(string.Concat("Total of ",
Convert.ToString(org.TotalDefectsIntroduced), " total defects introduced.\n"));
            statusTextBox.AppendText(string.Concat("Total of ",
Convert.ToString(org.TotalDefectsRemoved), " total defects removed.\n"));
            statusTextBox.ScrollToCaret();
        }

        private void noButton_Click(object sender, EventArgs e)
        {
            DisableAllButtons();
            queryTextBox.Clear();
            IAmPaused = false;
        }

        private void yesButton_Click(object sender, EventArgs e)
        {
            DisableAllButtons();
            queryTextBox.Clear();
            SimUpdate();
            IAmPaused = false;
        }

        private void SimUpdate()
        {
            updateForm1 = new UpdateForm();
            updateForm1.ShowDialog();

            if (updateForm1.ChangeUserStory.Equals(true))
            {
                storyName = org.StoryName();
                SelectStoryForm selectStoryForm1 = new SelectStoryForm(storyName);
                selectStoryForm1.ShowDialog();
                string[] selectedStory = selectStoryForm1.SelectedStory;
                int selectedCount = selectStoryForm1.SelectedCount;
                for (int i = 0; i < selectedCount; i++)
                {
                    string sName = selectedStory[i].Trim();
                    UserStory changedStory = org.GetStory(sName);
                    UpdateStoryForm updateStoryForm1 = new UpdateStoryForm(sName,
changedStory.EstimatedSize, changedStory.EstimatedDuration, changedStory.LocKnown,
org.EstimatedLocPerHour);
                    updateStoryForm1.ShowDialog();
```

```
            if (changedStory.Name == sName)
            {
                changedStory.UpdateSize(updateStoryForm1.LocAdded, updateStoryForm1.LocDeleted);
                changedStory.EstimatedDuration = updateStoryForm1.EstimatedDuration;
                changedStory.Update();
            }
        }
        MessageBox.Show("User stories Changed");
    }

    if (updateForm1.AddUserStory.Equals(true))
    {
        storyCount += updateForm1.AddCount;
        storyForm1 = new StoryForm("add", storyCount);
        storyForm1.FolderPath = folderPath;
        storyForm1.ShowDialog();
        org.UpdateStoryList(folderPath + "userstory.txt", storyCount);
        MessageBox.Show("User story Added");
    }

    if (updateForm1.RemoveUserStory.Equals(true))
    {
        storyCount -= updateForm1.RemoveCount;
        storyForm1 = new StoryForm("remove", storyCount);
        storyForm1.FolderPath = folderPath;
        storyForm1.ShowDialog();
        org.UpdateStoryList(folderPath + "userstory.txt", storyCount);
        MessageBox.Show("User story Removed");
    }

    if (updateForm1.UpdateStoryPriorityList.Equals(true))
    {
        organizationUpdate();
        MessageBox.Show("Priority Update");
    }

    storyName.Initialize();
    storyName = org.StoryName();
    storyListBox.Items.Clear();
    foreach (string item in storyName)
    {
        storyListBox.Items.Add(item);
    }
}

private void EnableAllButtons()
{
    yesButton.Enabled = true;
    noButton.Enabled = true;
    goStoryButton.Enabled = true;
    goAgentButton.Enabled = true;
    goTeamButton.Enabled = true;
```

```
}

private void DisableAllButtons()
{
   yesButton.Enabled = false;
   noButton.Enabled = false;
   goStoryButton.Enabled = false;
   goAgentButton.Enabled = false;
   goTeamButton.Enabled = false;
}

private void organizationUpdate()
{
   //Update Teams
   teamUpdateForm1 = new UpdateTeamForm(teamCount, storyCount);
   teamUpdateForm1.FolderPath = folderPath;
   //Add team names to the form
   #region Add team names to the form
   for (int i = 0; i < teamCount; i++)
   {
      teamUpdateForm1.AddTeam(org.Team[i].Name);
   }
   #endregion

   #region Add Stories to the form
   int j = -1;
   string fileName = folderPath + "userstory.txt";
   FileStream inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
   StreamReader sr = new StreamReader(inFile);
   string s = sr.ReadLine();
   while (s != null)
   {
      storyName[++j] = s;
      teamUpdateForm1.AddStory(s);
      s = sr.ReadLine();
   }
   sr.Close();
   inFile.Close();
   #endregion

   if (teamCount > 0)
   {
      teamUpdateForm1.ShowDialog();
      org.UpdateTeams(folderPath + "updateteams.txt", delimeter, teamCount);
   }

   summaryFile.WriteLine("Simulation and Organization updated");
   statusTextBox.AppendText("Simulation and Organization updated. \n");
   statusTextBox.ScrollToCaret();
}

private void goStoryButton_Click(object sender, EventArgs e)
{
```

```
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
               UserStory selectedStory = org.GetStory(selectedText);

               storyTextBox.AppendText(string.Concat("\nStory Name: ", selectedText, ". \n"));
               string measurement;
               for (int i = 0; i < measurementCount; i++)
               {
                  measurement = measurementList[i];
                  if (measurement.Equals("ComponentStatus"))
                  {
                     storyTextBox.AppendText(string.Concat("Active: ",
Convert.ToString(selectedStory.Active), ".\n"));
                     storyTextBox.AppendText(string.Concat("Completed Status: ",
Convert.ToString(selectedStory.Complete), ".\n"));
                     storyTextBox.AppendText(string.Concat("Current Activity: ", selectedStory.ActivityType,
".\n"));
                  }
                  if (measurement.Equals("PersonnelEffort"))
                  {
                     if (time >= 60)
                     {
                        storyTextBox.AppendText(string.Concat("Time Spent(in Person Hours): ",
Convert.ToString((int)(selectedStory.TimeSpent / 60)), " hours and ",
Convert.ToString(selectedStory.TimeSpent - 60 * (int)(selectedStory.TimeSpent / 60)), " mins. \n"));
                     }
                     else
                     {
                        storyTextBox.AppendText(string.Concat("Time Spent(in Person Hours): ",
Convert.ToString(selectedStory.TimeSpent), " mins. \n"));
                     }
                  }
                  if (measurement.Equals("LinesOfCode"))
                  {
                     storyTextBox.AppendText(string.Concat("LOC Planned: ",
Convert.ToString(selectedStory.LocPlanned), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Designed: ",
Convert.ToString(selectedStory.LocDesigned), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC DesignReviewed: ",
Convert.ToString(selectedStory.LocDesignReviewed), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Written: ",
Convert.ToString(selectedStory.LocWritten), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC CodeReviewed: ",
Convert.ToString(selectedStory.LocCodeReviewed), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Compiled: ",
Convert.ToString(selectedStory.LocCompiled), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Tested: ",
Convert.ToString(selectedStory.LocTested), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Refactored: ",
Convert.ToString(selectedStory.LocRefactored), ".\n"));
                     storyTextBox.AppendText(string.Concat("LOC Completed: ",
Convert.ToString(selectedStory.LocCompleted), ".\n"));
```

```
                }
                if (measurement.Equals("Defects"))
                {
                    storyTextBox.AppendText(string.Concat("Current Defects: ",
Convert.ToString(selectedStory.CurrentDefects), ".\n"));
                }
                storyTextBox.ScrollToCaret();
            }
        }
        else
            MessageBox.Show("Please Select a Team to view its stats", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }

    private void goAgentButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)agentListBox.SelectedItem;
        if (selectedText != null)
        {
            Agent selectedAgent = org.GetAgent(selectedText);
            agentTextBox.AppendText(string.Concat("Employee Name: ", selectedText, ". \n"));
            if (time >= 60)
            {
                agentTextBox.AppendText(string.Concat("Time: ", Convert.ToString((int)(time / 60)), " hours
and ", Convert.ToString(time - 60 * (int)(time / 60)), " mins.\n"));
            }
            else
            {
                agentTextBox.AppendText(string.Concat("Time: ", Convert.ToString(time), " mins. \n"));
            }
            agentTextBox.AppendText(string.Concat("Current Activity: ", selectedAgent.ActivityType,
".\n"));

            string measurement;
            for (int i = 0; i < measurementCount; i++)
            {
                measurement = measurementList[i];
                if (measurement.Equals("LinesOfCode"))
                {
                    agentTextBox.AppendText(string.Concat("LOC Planned: ",
Convert.ToString(selectedAgent.TotalLocPlanned), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Designed: ",
Convert.ToString(selectedAgent.TotalLocDesigned), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Written: ",
Convert.ToString(selectedAgent.TotalLocWritten), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Compiled: ",
Convert.ToString(selectedAgent.TotalLocCompiled), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Tested: ",
Convert.ToString(selectedAgent.TotalLocTested), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Refactored: ",
Convert.ToString(selectedAgent.TotalLocRefactored), ".\n"));
                    agentTextBox.AppendText(string.Concat("LOC Completed: ",
Convert.ToString(selectedAgent.TotalLocCompleted), ".\n"));
```

```
                }
                if (measurement.Equals("Defects"))
                {
                    agentTextBox.AppendText(string.Concat("Current Defects: ",
Convert.ToString(selectedAgent.CurrentDefects), ".\n"));
                    agentTextBox.AppendText(string.Concat("Total Defects Introduced: ",
Convert.ToString(selectedAgent.TotalDefectsIntroduced), ".\n"));
                    agentTextBox.AppendText(string.Concat("Total Defects Removed: ",
Convert.ToString(selectedAgent.TotalDefectsRemoved), ".\n\n\n"));
                }
                if (measurement.Equals("Productivity"))
                {
                    agentTextBox.AppendText(string.Concat("Productivity(loc/hour): ",
Convert.ToString(selectedAgent.TotalLocCompleted * 60 / time), ".\n"));
                }
                agentTextBox.ScrollToCaret();
            }
        }
        else
            MessageBox.Show("Please Select an Employee to view its stats", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    private void goTeamButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)teamListBox.SelectedItem;
        if (selectedText != null)
        {
            Team selectedTeam = org.GetTeam(selectedText);

            teamTextBox.AppendText(string.Concat("Team Name: ", selectedText, ". \n"));
            if (time >= 60)
            {
                teamTextBox.AppendText(string.Concat("Time: ", Convert.ToString((int)(time / 60)), " hours
and ", Convert.ToString(time - 60 * (int)(time / 60)), " mins. \n"));
            }
            else
            {
                teamTextBox.AppendText(string.Concat("Time: ", Convert.ToString(time), " mins. \n"));
            }

            teamTextBox.AppendText(string.Concat("Current Activity: ", selectedTeam.ActivityType,
".\n"));

            string measurement;
            for (int i = 0; i < measurementCount; i++)
            {
                measurement = measurementList[i];
                if (measurement.Equals("ComponentStatus"))
                {
                    teamTextBox.AppendText(string.Concat("Story Completed: ",
Convert.ToString(selectedTeam.StoryCompleted), ".\n"));
                }
```

```
                if (measurement.Equals("ProjectVelocity"))
                {
                    teamTextBox.AppendText(string.Concat("Project Velocity(story/hour): ",
Convert.ToString(selectedTeam.StoryCompleted * 60 / time), ".\n"));
                }
                if (measurement.Equals("EarnedValue"))
                {
                    teamTextBox.AppendText(string.Concat("User Story Completed: ",
Convert.ToString(selectedTeam.StoryCompleted), ".\n"));

                }
                if (measurement.Equals("LinesOfCode"))
                {
                    teamTextBox.AppendText(string.Concat("LOC Planned: ",
Convert.ToString(selectedTeam.LocPlanned), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Designed: ",
Convert.ToString(selectedTeam.LocDesigned), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Written: ",
Convert.ToString(selectedTeam.LocWritten), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Compiled: ",
Convert.ToString(selectedTeam.LocCompiled), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Tested: ",
Convert.ToString(selectedTeam.LocTested), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Refactored: ",
Convert.ToString(selectedTeam.LocRefactored), ".\n"));
                    teamTextBox.AppendText(string.Concat("LOC Completed: ",
Convert.ToString(selectedTeam.LocCompleted), ".\n"));
                }
                if (measurement.Equals("Defects"))
                {
                    teamTextBox.AppendText(string.Concat("Current Defects: ",
Convert.ToString(selectedTeam.CurrentDefects), ".\n"));
                    teamTextBox.AppendText(string.Concat("Total Defects Introduced: ",
Convert.ToString(selectedTeam.TotalDefectsIntroduced), ".\n"));
                    teamTextBox.AppendText(string.Concat("Total Defects Removed: ",
Convert.ToString(selectedTeam.TotalDefectsRemoved), ".\n\n\n"));
                }
                if (measurement.Equals("Productivity"))
                {
                    teamTextBox.AppendText(string.Concat("Productivity(loc/hour): ",
Convert.ToString(selectedTeam.LocCompleted * 60 / time), ".\n"));
                }
                teamTextBox.ScrollToCaret();
            }
        }
        else
            MessageBox.Show("Please Select a Team to view its stats", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);

    }

    private void DisplayOrganizationStatus()
    {
```

```
        if (time >= 60)
        {
            organizationTextBox.AppendText(string.Concat("Time: ", Convert.ToString((int)(time / 60)), "
hours and ", Convert.ToString(time - 60 * (int)(time / 60)), " mins. \n"));
        }
        else
        {
            organizationTextBox.AppendText(string.Concat("Time: ", Convert.ToString(time), " mins. \n"));
        }
        string measurement;
        for (int i = 0; i < measurementCount; i++)
        {
            measurement = measurementList[i];
            if (measurement.Equals("ComponentStatus"))
            {
                organizationTextBox.AppendText(string.Concat("Story Completed: ",
Convert.ToString(org.StoryCompleted), ".\n"));
            }
            if (measurement.Equals("ProjectVelocity"))
            {
                organizationTextBox.AppendText(string.Concat("Project Velocity(story/hour): ",
Convert.ToString(org.StoryCompleted * 60 / time), ".\n"));
            }
            if (measurement.Equals("EarnedValue"))
            {
                organizationTextBox.AppendText(string.Concat("User Story Completed: ",
Convert.ToString(org.StoryCompleted), ".\n"));
            }
            if (measurement.Equals("LinesOfCode"))
            {
                organizationTextBox.AppendText(string.Concat("LOC Planned: ",
Convert.ToString(org.LocPlanned), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Designed: ",
Convert.ToString(org.LocDesigned), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Written: ",
Convert.ToString(org.LocWritten), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Compiled: ",
Convert.ToString(org.LocCompiled), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Tested: ",
Convert.ToString(org.LocTested), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Refactored: ",
Convert.ToString(org.LocRefactored), ".\n"));
                organizationTextBox.AppendText(string.Concat("LOC Completed: ",
Convert.ToString(org.LocCompleted), ".\n"));
            }
            if (measurement.Equals("Defects"))
            {
                organizationTextBox.AppendText(string.Concat("Current Defects: ",
Convert.ToString(org.CurrentDefects), ".\n"));
                organizationTextBox.AppendText(string.Concat("Total Defects Introduced: ",
Convert.ToString(org.TotalDefectsIntroduced), ".\n"));
                organizationTextBox.AppendText(string.Concat("Total Defects Removed: ",
Convert.ToString(org.TotalDefectsRemoved), ".\n\n\n"));
```

```
            }
            if (measurement.Equals("Productivity"))
            {
                organizationTextBox.AppendText(string.Concat("Productivity(loc/hour): ",
Convert.ToString(org.LocCompleted * 60 / time), ".\n"));
            }
            organizationTextBox.ScrollToCaret();
        }
    }

    private void DisplayExpectedValues()
    {
        if (time >= 60)
        {
            expectedTextBox.Text = string.Concat("Time: ", Convert.ToString((int)(time / 60)), " hours and
", Convert.ToString(time - 60 * (int)(time / 60)), " mins. \n");
        }
        else
        {
            expectedTextBox.Text = string.Concat("Time: ", Convert.ToString(time), " mins. \n");
        }

        expectedTextBox.AppendText("Stories Completed: \n");
        foreach (string item in storyName)
        {
            if (item != null)
            {
                UserStory story = org.GetStory(item);
                if ((story.Active.Equals(true)) && (story.TimeSpent >= story.EstimatedDuration) &&
(story.EstimatedSize > 0))
                {
                    expectedTextBox.AppendText(string.Concat("Story: ", story.Name, ", Duration: ",
story.EstimatedDuration, " mins.\n"));
                }
            }
        }

        expectedTextBox.ScrollToCaret();
    }

    private void DisplayActualValues()
    {
        if (time >= 60)
        {
            actualTextBox.Text = string.Concat("Time: ", Convert.ToString((int)(time / 60)), " hour(s) and ",
Convert.ToString(time - 60 * (int)(time / 60)), " min(s). \n");
        }
        else
        {
            actualTextBox.Text = string.Concat("Time: ", Convert.ToString(time), " min(s). \n");
        }

        actualTextBox.AppendText("Stories Completed: \n");
```

```
            foreach (string item in storyName)
            {
                if (item != null)
                {
                    UserStory story = org.GetStory(item);
                    if ((story.Active.Equals(true)) && (story.Complete.Equals(true)) && (story.EstimatedSize >
0))
                    {
                        actualTextBox.AppendText(string.Concat("Story: ", story.Name, ", Duration: ",
story.TimeSpent, " mins. \n"));
                    }
                }
            }
        }

        private void aboutButton_Click(object sender, EventArgs e)
        {
            AboutForm aboutForm1 = new AboutForm();
            aboutForm1.Show();
        }
    }
}
namespace SDPTool
{
    partial class SelectStoryForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
```

```
            this.label2 = new System.Windows.Forms.Label();
            this.storyListBox = new System.Windows.Forms.ListBox();
            this.removeStoryButton = new System.Windows.Forms.Button();
            this.label8 = new System.Windows.Forms.Label();
            this.selectedStoryListBox = new System.Windows.Forms.ListBox();
            this.selectStoryButton = new System.Windows.Forms.Button();
            this.submitButton = new System.Windows.Forms.Button();
            this.label4 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(56, 46);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(79, 13);
            this.label2.TabIndex = 84;
            this.label2.Text = "Current Stories:";
            //
            // storyListBox
            //
            this.storyListBox.FormattingEnabled = true;
            this.storyListBox.Location = new System.Drawing.Point(59, 62);
            this.storyListBox.Name = "storyListBox";
            this.storyListBox.Size = new System.Drawing.Size(118, 121);
            this.storyListBox.TabIndex = 85;
            //
            // removeStoryButton
            //
            this.removeStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.removeStoryButton.Location = new System.Drawing.Point(183, 114);
            this.removeStoryButton.Name = "removeStoryButton";
            this.removeStoryButton.Size = new System.Drawing.Size(39, 38);
            this.removeStoryButton.TabIndex = 81;
            this.removeStoryButton.Text = "<<";
            this.removeStoryButton.UseVisualStyleBackColor = true;
            this.removeStoryButton.Click += new System.EventHandler(this.removeStoryButton_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(225, 46);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(112, 13);
            this.label8.TabIndex = 82;
            this.label8.Text = "Selected User Stories:";
            //
            // selectedStoryListBox
            //
            this.selectedStoryListBox.FormattingEnabled = true;
            this.selectedStoryListBox.Location = new System.Drawing.Point(228, 62);
            this.selectedStoryListBox.Name = "selectedStoryListBox";
```

```
            this.selectedStoryListBox.Size = new System.Drawing.Size(118, 121);
            this.selectedStoryListBox.TabIndex = 83;
            //
            // selectStoryButton
            //
            this.selectStoryButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.selectStoryButton.Location = new System.Drawing.Point(183, 70);
            this.selectStoryButton.Name = "selectStoryButton";
            this.selectStoryButton.Size = new System.Drawing.Size(39, 38);
            this.selectStoryButton.TabIndex = 80;
            this.selectStoryButton.Text = ">>";
            this.selectStoryButton.UseVisualStyleBackColor = true;
            this.selectStoryButton.Click += new System.EventHandler(this.selectStoryButton_Click);
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(251, 194);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 79;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(12, 9);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(402, 17);
            this.label4.TabIndex = 86;
            this.label4.Text = "Please select all  the user stories you want to change:";
            //
            // SelectStoryForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(418, 237);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.storyListBox);
            this.Controls.Add(this.removeStoryButton);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.selectedStoryListBox);
            this.Controls.Add(this.selectStoryButton);
            this.Controls.Add(this.submitButton);
            this.Name = "SelectStoryForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
```

```
            this.Text = "SelectStoryForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox storyListBox;
        private System.Windows.Forms.Button removeStoryButton;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.ListBox selectedStoryListBox;
        private System.Windows.Forms.Button selectStoryButton;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Label label4;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SDPTool
{
   public partial class SelectStoryForm : Form
   {
      static int TOTALSTORY = 50;
      string[] selectedStory = new string[TOTALSTORY];
      int selectedCount = 0;

      public string[] SelectedStory
      {
         get  //get accessor method
         {
            return selectedStory;
         }
      }
      public int SelectedCount
      {
         get  //get accessor method
         {
            return selectedCount;
         }
      }

      public SelectStoryForm(string[] storyName)
      {
         InitializeComponent();
         storyListBox.Items.Clear();
         foreach (string item in storyName)
         {
            storyListBox.Items.Add(item);
         }
      }

      private void submitButton_Click(object sender, EventArgs e)
      {
         int i = 0;
         selectedCount = 0;
         foreach (object item in selectedStoryListBox.Items)
         {
            selectedStory[i++] = (string)item;
            selectedCount++;
         }
         Close();
      }
```

```
        private void selectStoryButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)storyListBox.SelectedItem;
            if (selectedText != null)
            {
                selectedStoryListBox.Items.Add(selectedText);
                storyListBox.Items.Remove(storyListBox.SelectedItem);
                removeStoryButton.Enabled = true;
            }
            else
            {
                MessageBox.Show("Please Select a story", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void removeStoryButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedStoryListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedStoryListBox.Items.Remove(selectedItem);
                storyListBox.Items.Add(selectedItem);
                if (selectedStoryListBox.Items.Count == 0)
                    removeStoryButton.Enabled = false;
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace SDPTool
{
   static class Program
   {
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new SimulationForm());
      }
   }
}
```

```
using System;
using System.IO;
using System.Threading;
using System.Windows.Forms;

// ******Note: 1 timeStep = 1 minute in real time
// ******Simulation pauses at each interval which is entered in minutes of real time.

namespace SDPTool
{
   class Organization
   {
      static int MAX_AGENTS = 50;
      static int MAX_TEAMS = MAX_AGENTS / 2;
      static int TOTALSTORY = 50;
      static int TOTALWORDS = (5 + TOTALSTORY);
      int TOTALAGENTS = 0;
      Agent[] agent = new Agent[MAX_AGENTS];
      Thread[] agentThread = new Thread[MAX_AGENTS];
      Team[] team = new Team[MAX_TEAMS];
      Thread[] teamThread = new Thread[MAX_TEAMS];
      UserStory[] userStory = new UserStory[TOTALSTORY];

      int agentCount;
      int teamCount;
      int storyCount = 0;
      int storyCompleted = 0;
      string folderPath;
      FileStream inFile;
      StreamReader sr;
      File summaryFile;
      double time;
      double b0 = 0;
      double b1 = 0;
      double estimatedLocPerHour = 0;
      double locPlanned = 0;
      double locDesigned = 0;
      double locDesignReviewed = 0;
      double locWritten = 0;
      double locCodeReviewed = 0;
      double locCompiled = 0;
      double locTested = 0;
      double locRefactored = 0;
      double locCompleted = 0;
      double currentDefects;
      double totalDefectsRemoved = 0;
      double totalDefectsIntroduced = 0;
      double projectSize = 0;

      public string FolderPath
      {
         get  //get accessor method
         {
```

```
        return folderPath;
    }
    set   //set accessor method
    {
        folderPath = value;
    }
}

public double Time
{
    get  //get accessor method
    {
        return time;
    }
    set   //set accessor method
    {
        time = value;
    }
}

public double ProjectSize
{
    get  //get accessor method
    {
        return projectSize;
    }
    set   //set accessor method
    {
        projectSize = value;
    }
}

public double EstimationError
{
    get  //get accessor method
    {
        return b1;
    }
    set   //set accessor method
    {
        b1 = value;
    }
}

public double EstimatedLocPerHour
{
    get  //get accessor method
    {
        return estimatedLocPerHour;
    }
    set   //set accessor method
    {
        estimatedLocPerHour = value;
```

```
    }
}

public double LocPlanned
{
    get  //get accessor method
    {
        return locPlanned;
    }
    set   //set accessor method
    {
        locPlanned = value;
    }
}

public double LocDesigned
{
    get  //get accessor method
    {
        return locDesigned;
    }
    set   //set accessor method
    {
        locDesigned = value;
    }
}

public double LocDesignReviewed
{
    get  //get accessor method
    {
        return locDesignReviewed;
    }
    set   //set accessor method
    {
        locDesignReviewed = value;
    }
}

public double LocWritten
{
    get  //get accessor method
    {
        return locWritten;
    }
    set   //set accessor method
    {
        locWritten = value;
    }
}

public double LocCodeReviewed
{
```

```
    get  //get accessor method
    {
        return locCodeReviewed;
    }
    set   //set accessor method
    {
        locCodeReviewed = value;
    }
}

public double LocCompiled
{
    get  //get accessor method
    {
        return locCompiled;
    }
    set   //set accessor method
    {
        locCompiled = value;
    }
}

public double LocTested
{
    get  //get accessor method
    {
        return locTested;
    }
    set   //set accessor method
    {
        locTested = value;
    }
}

public double LocRefactored
{
    get  //get accessor method
    {
        return locRefactored;
    }
    set   //set accessor method
    {
        locRefactored = value;
    }
}

public double LocCompleted
{
    get  //get accessor method
    {
        return locCompleted;
    }
    set   //set accessor method
```

```
    {
        locCompleted = value;
    }
}

public int StoryCompleted
{
    get  //get accessor method
    {
        return storyCompleted;
    }
    set   //set accessor method
    {
        storyCompleted = value;
    }
}

public int StoryCount
{
    get  //get accessor method
    {
        return storyCount;
    }
    set   //set accessor method
    {
        storyCount = value;
    }
}

public double CurrentDefects
{
    get  //get accessor method
    {
        return currentDefects;
    }
    set   //set accessor method
    {
        currentDefects = value;
    }
}

public double TotalDefectsIntroduced
{
    get  //get accessor method
    {
        return totalDefectsIntroduced;
    }
    set   //set accessor method
    {
        totalDefectsIntroduced = value;
    }
}
```

```
public double TotalDefectsRemoved
{
   get  //get accessor method
   {
      return totalDefectsRemoved;
   }
   set   //set accessor method
   {
      totalDefectsRemoved = value;
   }
}
public Agent[] Agent
{
   get
   {
      return agent;
   }
}

public Team[] Team
{
   get
   {
      return team;
   }
}

public int AgentCount
{
   get
   {
      return agentCount;
   }
   set
   {
      agentCount = value;
   }
}

public int TeamCount
{
   get
   {
      return teamCount;
   }
   set
   {
      teamCount = value;
   }
}

public string[] StoryName()
{
```

```
            string[] sNames = new string[storyCount];
            for (int i = 0; i < storyCount; i++)
            {
                sNames[i] = userStory[i].Name;
            }
            return sNames;
        }

        public Agent GetAgent(string name)
        {
            for (int i = 0; i < agentCount; i++)
            {
                if (agent[i].Name == name)
                    return agent[i];
            }
            MessageBox.Show("Can't find an agent", "Closing the simulation", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            Environment.Exit(0);
            return agent[0];
        }

        public Team GetTeam(string name)
        {
            for (int i = 0; i < teamCount; i++)
            {
                if (team[i].Name == name)
                    return team[i];
            }
            MessageBox.Show("Can't find a team", "Closing the simulation", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            Environment.Exit(0);
            return team[0];
        }

        public UserStory GetStory(string teamName, string name)
        {
            for (int i = 0; i < storyCount; i++)
            {
                if (userStory[i].Name == name)
                {
                    if (userStory[i].EstimatedSize <= 0)
                    {
                        StoryEstimateForm storyEstimateForm1 = new StoryEstimateForm(teamName,
userStory[i].Name);
                        storyEstimateForm1.ShowDialog();
                        userStory[i].EstimatedSize = storyEstimateForm1.EstimatedSize;
                        userStory[i].EstimatedDuration = storyEstimateForm1.EstimatedDuration;
                        userStory[i].LocKnown = storyEstimateForm1.LocKnown;
                        if (userStory[i].EstimatedSize == -1)
                        {
                            userStory[i].EstimatedSize = (userStory[i].EstimatedDuration) * estimatedLocPerHour /
60;
                        }
```

```
            }
            return userStory[i];
          }
        }
        MessageBox.Show("Can't find the user story", "Closing the simulation", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        Environment.Exit(0);
        return userStory[0];    //If no match found, returns the first story!
      }

      public UserStory GetStory(string name)
      {
        for (int i = 0; i < storyCount; i++)
        {
          if (userStory[i].Name == name)
          {
            return userStory[i];
          }
        }
        MessageBox.Show("Can't find the user story", "Closing the simulation", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        Environment.Exit(0);
        return userStory[0];  //If no match found, returns the first story!
      }

      public void CreateStories(string[] sNames)
      {
        foreach (string item in sNames)
        {
          if (item != null)
          {
            userStory[++storyCount - 1] = new UserStory(storyCount - 1, item, 0, 0);
          }
        }
      }

      public void CreateAgents(string fileName, char[] delimeter, int aCount)
      {
        TOTALAGENTS = aCount;
        agentCount = 0;
        locWritten = 0;

        inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
        sr = new StreamReader(inFile);
        string s = sr.ReadLine();
        while ((s != null) && (agentCount < TOTALAGENTS))
        {
          agent[++agentCount - 1] = new Agent(s, delimeter);
          b0 += agent[agentCount - 1].PspB0;
          b1 += agent[agentCount - 1].PspB1;
          estimatedLocPerHour += (agent[agentCount - 1].PspLowerLocPerHour + agent[agentCount -
1].PspUpperLocPerHour) / 2;
          s = sr.ReadLine();
```

```
    }
    b0 = b0 / agentCount;
    b1 = b1 / agentCount;
    estimatedLocPerHour = estimatedLocPerHour / agentCount;
    sr.Close();
    inFile.Close();
}

public void CreateTeams(string fileName, char[] delimeter, int tCount)
{
    teamCount = 0;
    inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
    sr = new StreamReader(inFile);
    string s = sr.ReadLine();

    while ((s != null) && (teamCount < tCount))
    {
        string[] features = new string[TOTALWORDS + 1];
        string[] words = new string[TOTALWORDS];

        words = s.Split(delimeter);

        int i = 0;
        foreach (string word in words)
        {
            features[i++] = word.Trim();
        }

        if ((i > 0) && (i <= TOTALWORDS))
        {
            int index = -1;
            string[] sNames = new string[i - 3];
            int id = ++teamCount - 1;
            string name = features[0];
            string member1 = features[1];
            string member2 = features[2];
            while (++index + 3 < i - 1)
            {
                sNames[index] = features[index + 3];
            }

            int sCount = Int32.Parse(features[i - 1]);

            team[id] = new Team(id, name, GetAgent(member1), GetAgent(member2), sCount);
            team[id].StoryNames = sNames;
            team[id].Parent = this;
            team[id].FolderPath = FolderPath;
            team[id].B0 = b0;
            team[id].B1 = b1;
            team[id].EstimatedLocPerHour = estimatedLocPerHour;
        }
        else
        {
```

```
                // Stop the simulation ...
                // Error in teams.txt
                MessageBox.Show("Error in teams.txt", "Closing the simulation", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                Environment.Exit(0);
            }
            s = sr.ReadLine();
        }
        sr.Close();
        inFile.Close();
    }

    public void UpdateTeams(string fileName, char[] delimeter, int tCount)
    {
        inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
        sr = new StreamReader(inFile);
        string s = sr.ReadLine();

        while ((s != null))
        {
            string[] features = new string[TOTALWORDS + 1];
            string[] words = new string[TOTALWORDS];

            words = s.Split(delimeter);

            int i = 0;
            foreach (string word in words)
            {
                features[i++] = word.Trim();
            }

            if ((i > 0) && (i <= TOTALWORDS))
            {
                int index = -1;
                string[] sNames = new string[Int32.Parse(features[i - 1])];//[i - 2];
                string name = features[0];
                Team teamToUpdate = GetTeam(name);
                while (++index < i - 4)
                {
                    sNames[index] = features[index + 1];
                    bool newStory = true;
                    for (int k = 0; k < storyCount; k++)
                    {
                        if (userStory[k].Name == sNames[index])
                        {
                            newStory = false;
                            break;
                        }
                    }
                    if (newStory.Equals(true))
                    {
                        userStory[++storyCount - 1] = new UserStory(storyCount - 1, sNames[index], 0, 0);
                    }
```

```
                }

                int sCount = Int32.Parse(features[i - 1]);
                teamToUpdate.StoryCount = sCount;
                teamToUpdate.StoryNames = sNames;
                teamToUpdate.StoryIndex = 0;
                teamToUpdate.StartStory = true;
            }
            else
            {
                // Stop the simulation ...
                // Error in teams.txt
                MessageBox.Show("Error in updateteams.txt", "Closing the simulation",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                Environment.Exit(0);
            }
            s = sr.ReadLine();
        }
        sr.Close();
        inFile.Close();
    }

    public void UpdateStoryList(string fName, int count)
    {
        //make all user stories inactive
        for (int k = 0; k < storyCount; k++)
        {
            userStory[k].Active = false;
        }

        //add all the new user stories and activate the user stories
        int sCount = 0;
        string fileName = fName;
        FileStream inFile = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
        StreamReader sr = new StreamReader(inFile);
        string s = sr.ReadLine();
        while ((s != null) && (sCount < count))
        {
            s = s.Trim();
            bool newStory = true;
            for (int k = 0; k < storyCount; k++)
            {
                if (userStory[k].Name == s)
                {
                    newStory = false;
                    userStory[k].Active = true;
                    break;
                }
            }
            if (newStory.Equals(true))
            {
                userStory[++storyCount - 1] = new UserStory(storyCount - 1, s, 0, 0);
            }
```

```
      s = sr.ReadLine();
      sCount++;
    }
    sr.Close();
    inFile.Close();
}

public void Summarize()
{
    locPlanned = 0;
    locDesigned = 0;
    locDesignReviewed = 0;
    locWritten = 0;
    locCodeReviewed = 0;
    locCompiled = 0;
    locTested = 0;
    locRefactored = 0;
    locCompleted = 0;
    storyCompleted = 0;
    currentDefects = 0;
    totalDefectsIntroduced = 0;
    totalDefectsRemoved = 0;

    for (int i = 0; i < teamCount; i++)
    {
        locPlanned += team[i].LocPlanned;
        locDesigned += team[i].LocDesigned;
        locDesignReviewed += team[i].LocDesignReviewed;
        locWritten += team[i].LocWritten;
        locCodeReviewed += team[i].LocCodeReviewed;
        locCompiled += team[i].LocCompiled;
        locTested += team[i].LocTested;
        locRefactored += team[i].LocRefactored;
        locCompleted += team[i].LocCompleted;
        storyCompleted += team[i].StoryCompleted;
        currentDefects += team[i].CurrentDefects;
        totalDefectsIntroduced += team[i].TotalDefectsIntroduced;
        totalDefectsRemoved += team[i].TotalDefectsRemoved;
    }
    if (currentDefects < 0)
    {
        currentDefects = 0;
        for (int i = 0; i < teamCount; i++)
        {
            team[i].CurrentDefects = 0;
        }
    }
    if (totalDefectsRemoved > totalDefectsIntroduced)
        totalDefectsRemoved = totalDefectsIntroduced;
}

public void Run()
{
```

```
            //Run each team...
            for (int i = 0; i < teamCount; i++)
            {
               teamThread[i] = new Thread(new ThreadStart(team[i].Run));
               teamThread[i].Start();
            }

            //block the current thread
            // Use the Join method to block the current thread
            // until the object's thread terminates.
            for (int i = 0; i < teamCount; i++)
            {
               teamThread[i].Join();
            }

            //Sumarize each agents performance
            Summarize();
         }

         public Organization(string agentFile, char[] delimeter, int aCount)
         {
            summaryFile = new File(folderPath + "summary.txt");
            CreateAgents(agentFile, delimeter, aCount);
         }
      }
   }
```

```
namespace SDPTool
{
   partial class MeasurementForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.submitButton = new System.Windows.Forms.Button();
         this.scheduleRemoveButton = new System.Windows.Forms.Button();
         this.label3 = new System.Windows.Forms.Label();
         this.label2 = new System.Windows.Forms.Label();
         this.selectedScheduleListBox = new System.Windows.Forms.ListBox();
         this.scheduleSelectButton = new System.Windows.Forms.Button();
         this.scheduleListBox = new System.Windows.Forms.ListBox();
         this.label1 = new System.Windows.Forms.Label();
         this.label4 = new System.Windows.Forms.Label();
         this.selectedMeasurementListBox = new System.Windows.Forms.ListBox();
         this.ScheduleGroupBox = new System.Windows.Forms.GroupBox();
         this.resourcesGroupBox = new System.Windows.Forms.GroupBox();
         this.resourcesListBox = new System.Windows.Forms.ListBox();
         this.resourcesSelectButton = new System.Windows.Forms.Button();
         this.selectedResourcesListBox = new System.Windows.Forms.ListBox();
         this.label5 = new System.Windows.Forms.Label();
         this.label6 = new System.Windows.Forms.Label();
         this.resourcesRemoveButton = new System.Windows.Forms.Button();
         this.label7 = new System.Windows.Forms.Label();
         this.qualityGroupBox = new System.Windows.Forms.GroupBox();
         this.qualityListBox = new System.Windows.Forms.ListBox();
```

```
            this.qualitySelectButton = new System.Windows.Forms.Button();
            this.selectedQualityListBox = new System.Windows.Forms.ListBox();
            this.label8 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.qualityRemoveButton = new System.Windows.Forms.Button();
            this.label10 = new System.Windows.Forms.Label();
            this.sizeGroupBox = new System.Windows.Forms.GroupBox();
            this.sizeListBox = new System.Windows.Forms.ListBox();
            this.sizeSelectButton = new System.Windows.Forms.Button();
            this.selectedSizeListBox = new System.Windows.Forms.ListBox();
            this.label11 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.sizeRemoveButton = new System.Windows.Forms.Button();
            this.label13 = new System.Windows.Forms.Label();
            this.performanceGroupBox = new System.Windows.Forms.GroupBox();
            this.performanceListBox = new System.Windows.Forms.ListBox();
            this.performanceSelectButton = new System.Windows.Forms.Button();
            this.selectedPerformanceListBox = new System.Windows.Forms.ListBox();
            this.label14 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.performanceRemoveButton = new System.Windows.Forms.Button();
            this.label16 = new System.Windows.Forms.Label();
            this.ScheduleGroupBox.SuspendLayout();
            this.resourcesGroupBox.SuspendLayout();
            this.qualityGroupBox.SuspendLayout();
            this.sizeGroupBox.SuspendLayout();
            this.performanceGroupBox.SuspendLayout();
            this.SuspendLayout();
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(857, 305);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(95, 29);
            this.submitButton.TabIndex = 33;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // scheduleRemoveButton
            //
            this.scheduleRemoveButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.scheduleRemoveButton.Location = new System.Drawing.Point(130, 109);
            this.scheduleRemoveButton.Name = "scheduleRemoveButton";
            this.scheduleRemoveButton.Size = new System.Drawing.Size(39, 38);
            this.scheduleRemoveButton.TabIndex = 23;
            this.scheduleRemoveButton.Text = "<<";
            this.scheduleRemoveButton.UseVisualStyleBackColor = true;
            this.scheduleRemoveButton.Click += new
System.EventHandler(this.scheduleRemoveButton_Click);
```

```
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(172, 27);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(124, 13);
            this.label3.TabIndex = 24;
            this.label3.Text = "Selected Measurements:";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(6, 30);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(125, 13);
            this.label2.TabIndex = 20;
            this.label2.Text = "Available Measurements:";
            //
            // selectedScheduleListBox
            //
            this.selectedScheduleListBox.FormattingEnabled = true;
            this.selectedScheduleListBox.Location = new System.Drawing.Point(175, 46);
            this.selectedScheduleListBox.Name = "selectedScheduleListBox";
            this.selectedScheduleListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedScheduleListBox.TabIndex = 25;
            //
            // scheduleSelectButton
            //
            this.scheduleSelectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.scheduleSelectButton.Location = new System.Drawing.Point(130, 55);
            this.scheduleSelectButton.Name = "scheduleSelectButton";
            this.scheduleSelectButton.Size = new System.Drawing.Size(39, 38);
            this.scheduleSelectButton.TabIndex = 22;
            this.scheduleSelectButton.Text = ">>";
            this.scheduleSelectButton.UseVisualStyleBackColor = true;
            this.scheduleSelectButton.Click += new System.EventHandler(this.scheduleSelectButton_Click);
            //
            // scheduleListBox
            //
            this.scheduleListBox.FormattingEnabled = true;
            this.scheduleListBox.Location = new System.Drawing.Point(6, 46);
            this.scheduleListBox.Name = "scheduleListBox";
            this.scheduleListBox.Size = new System.Drawing.Size(118, 108);
            this.scheduleListBox.TabIndex = 21;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(6, 7);
```

```
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(182, 17);
            this.label1.TabIndex = 34;
            this.label1.Text = "Schedule and Progress:";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(663, 180);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(186, 17);
            this.label4.TabIndex = 35;
            this.label4.Text = "Selected Measurements:";
            this.label4.Visible = false;
            //
            // selectedMeasurementListBox
            //
            this.selectedMeasurementListBox.FormattingEnabled = true;
            this.selectedMeasurementListBox.Location = new System.Drawing.Point(677, 200);
            this.selectedMeasurementListBox.Name = "selectedMeasurementListBox";
            this.selectedMeasurementListBox.Size = new System.Drawing.Size(148, 134);
            this.selectedMeasurementListBox.TabIndex = 36;
            this.selectedMeasurementListBox.Visible = false;
            //
            // ScheduleGroupBox
            //
            this.ScheduleGroupBox.Controls.Add(this.scheduleListBox);
            this.ScheduleGroupBox.Controls.Add(this.scheduleSelectButton);
            this.ScheduleGroupBox.Controls.Add(this.selectedScheduleListBox);
            this.ScheduleGroupBox.Controls.Add(this.label2);
            this.ScheduleGroupBox.Controls.Add(this.label3);
            this.ScheduleGroupBox.Controls.Add(this.scheduleRemoveButton);
            this.ScheduleGroupBox.Controls.Add(this.label1);
            this.ScheduleGroupBox.Location = new System.Drawing.Point(10, 12);
            this.ScheduleGroupBox.Name = "ScheduleGroupBox";
            this.ScheduleGroupBox.Size = new System.Drawing.Size(311, 162);
            this.ScheduleGroupBox.TabIndex = 44;
            this.ScheduleGroupBox.TabStop = false;
            //
            // resourcesGroupBox
            //
            this.resourcesGroupBox.Controls.Add(this.resourcesListBox);
            this.resourcesGroupBox.Controls.Add(this.resourcesSelectButton);
            this.resourcesGroupBox.Controls.Add(this.selectedResourcesListBox);
            this.resourcesGroupBox.Controls.Add(this.label5);
            this.resourcesGroupBox.Controls.Add(this.label6);
            this.resourcesGroupBox.Controls.Add(this.resourcesRemoveButton);
            this.resourcesGroupBox.Controls.Add(this.label7);
            this.resourcesGroupBox.Location = new System.Drawing.Point(333, 12);
            this.resourcesGroupBox.Name = "resourcesGroupBox";
            this.resourcesGroupBox.Size = new System.Drawing.Size(311, 162);
```

```
            this.resourcesGroupBox.TabIndex = 45;
            this.resourcesGroupBox.TabStop = false;
            //
            // resourcesListBox
            //
            this.resourcesListBox.FormattingEnabled = true;
            this.resourcesListBox.Location = new System.Drawing.Point(6, 46);
            this.resourcesListBox.Name = "resourcesListBox";
            this.resourcesListBox.Size = new System.Drawing.Size(118, 108);
            this.resourcesListBox.TabIndex = 21;
            //
            // resourcesSelectButton
            //
            this.resourcesSelectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.resourcesSelectButton.Location = new System.Drawing.Point(130, 55);
            this.resourcesSelectButton.Name = "resourcesSelectButton";
            this.resourcesSelectButton.Size = new System.Drawing.Size(39, 38);
            this.resourcesSelectButton.TabIndex = 22;
            this.resourcesSelectButton.Text = ">>";
            this.resourcesSelectButton.UseVisualStyleBackColor = true;
            this.resourcesSelectButton.Click += new System.EventHandler(this.resourcesSelectButton_Click);
            //
            // selectedResourcesListBox
            //
            this.selectedResourcesListBox.FormattingEnabled = true;
            this.selectedResourcesListBox.Location = new System.Drawing.Point(175, 46);
            this.selectedResourcesListBox.Name = "selectedResourcesListBox";
            this.selectedResourcesListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedResourcesListBox.TabIndex = 25;
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(6, 30);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(125, 13);
            this.label5.TabIndex = 20;
            this.label5.Text = "Available Measurements:";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(172, 27);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(124, 13);
            this.label6.TabIndex = 24;
            this.label6.Text = "Selected Measurements:";
            //
            // resourcesRemoveButton
            //
            this.resourcesRemoveButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.resourcesRemoveButton.Location = new System.Drawing.Point(130, 109);
            this.resourcesRemoveButton.Name = "resourcesRemoveButton";
            this.resourcesRemoveButton.Size = new System.Drawing.Size(39, 38);
            this.resourcesRemoveButton.TabIndex = 23;
            this.resourcesRemoveButton.Text = "<<";
            this.resourcesRemoveButton.UseVisualStyleBackColor = true;
            this.resourcesRemoveButton.Click += new
System.EventHandler(this.resourcesRemoveButton_Click);
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(6, 7);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(159, 17);
            this.label7.TabIndex = 34;
            this.label7.Text = "Resources and Cost:";
            //
            // qualityGroupBox
            //
            this.qualityGroupBox.Controls.Add(this.qualityListBox);
            this.qualityGroupBox.Controls.Add(this.qualitySelectButton);
            this.qualityGroupBox.Controls.Add(this.selectedQualityListBox);
            this.qualityGroupBox.Controls.Add(this.label8);
            this.qualityGroupBox.Controls.Add(this.label9);
            this.qualityGroupBox.Controls.Add(this.qualityRemoveButton);
            this.qualityGroupBox.Controls.Add(this.label10);
            this.qualityGroupBox.Location = new System.Drawing.Point(333, 180);
            this.qualityGroupBox.Name = "qualityGroupBox";
            this.qualityGroupBox.Size = new System.Drawing.Size(311, 162);
            this.qualityGroupBox.TabIndex = 47;
            this.qualityGroupBox.TabStop = false;
            //
            // qualityListBox
            //
            this.qualityListBox.FormattingEnabled = true;
            this.qualityListBox.Location = new System.Drawing.Point(6, 46);
            this.qualityListBox.Name = "qualityListBox";
            this.qualityListBox.Size = new System.Drawing.Size(118, 108);
            this.qualityListBox.TabIndex = 21;
            //
            // qualitySelectButton
            //
            this.qualitySelectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.qualitySelectButton.Location = new System.Drawing.Point(130, 55);
            this.qualitySelectButton.Name = "qualitySelectButton";
            this.qualitySelectButton.Size = new System.Drawing.Size(39, 38);
            this.qualitySelectButton.TabIndex = 22;
            this.qualitySelectButton.Text = ">>";
            this.qualitySelectButton.UseVisualStyleBackColor = true;
```

```
            this.qualitySelectButton.Click += new System.EventHandler(this.qualitySelectButton_Click);
            //
            // selectedQualityListBox
            //
            this.selectedQualityListBox.FormattingEnabled = true;
            this.selectedQualityListBox.Location = new System.Drawing.Point(175, 46);
            this.selectedQualityListBox.Name = "selectedQualityListBox";
            this.selectedQualityListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedQualityListBox.TabIndex = 25;
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(6, 30);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(125, 13);
            this.label8.TabIndex = 20;
            this.label8.Text = "Available Measurements:";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Location = new System.Drawing.Point(172, 27);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(124, 13);
            this.label9.TabIndex = 24;
            this.label9.Text = "Selected Measurements:";
            //
            // qualityRemoveButton
            //
            this.qualityRemoveButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.qualityRemoveButton.Location = new System.Drawing.Point(130, 109);
            this.qualityRemoveButton.Name = "qualityRemoveButton";
            this.qualityRemoveButton.Size = new System.Drawing.Size(39, 38);
            this.qualityRemoveButton.TabIndex = 23;
            this.qualityRemoveButton.Text = "<<";
            this.qualityRemoveButton.UseVisualStyleBackColor = true;
            this.qualityRemoveButton.Click += new System.EventHandler(this.qualityRemoveButton_Click);
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label10.Location = new System.Drawing.Point(6, 7);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(125, 17);
            this.label10.TabIndex = 34;
            this.label10.Text = "Product Quality:";
            //
            // sizeGroupBox
            //
```

```
            this.sizeGroupBox.Controls.Add(this.sizeListBox);
            this.sizeGroupBox.Controls.Add(this.sizeSelectButton);
            this.sizeGroupBox.Controls.Add(this.selectedSizeListBox);
            this.sizeGroupBox.Controls.Add(this.label11);
            this.sizeGroupBox.Controls.Add(this.label12);
            this.sizeGroupBox.Controls.Add(this.sizeRemoveButton);
            this.sizeGroupBox.Controls.Add(this.label13);
            this.sizeGroupBox.Location = new System.Drawing.Point(10, 180);
            this.sizeGroupBox.Name = "sizeGroupBox";
            this.sizeGroupBox.Size = new System.Drawing.Size(311, 162);
            this.sizeGroupBox.TabIndex = 46;
            this.sizeGroupBox.TabStop = false;
            //
            // sizeListBox
            //
            this.sizeListBox.FormattingEnabled = true;
            this.sizeListBox.Location = new System.Drawing.Point(6, 46);
            this.sizeListBox.Name = "sizeListBox";
            this.sizeListBox.Size = new System.Drawing.Size(118, 108);
            this.sizeListBox.TabIndex = 21;
            //
            // sizeSelectButton
            //
            this.sizeSelectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.sizeSelectButton.Location = new System.Drawing.Point(130, 55);
            this.sizeSelectButton.Name = "sizeSelectButton";
            this.sizeSelectButton.Size = new System.Drawing.Size(39, 38);
            this.sizeSelectButton.TabIndex = 22;
            this.sizeSelectButton.Text = ">>";
            this.sizeSelectButton.UseVisualStyleBackColor = true;
            this.sizeSelectButton.Click += new System.EventHandler(this.sizeSelectButton_Click);
            //
            // selectedSizeListBox
            //
            this.selectedSizeListBox.FormattingEnabled = true;
            this.selectedSizeListBox.Location = new System.Drawing.Point(175, 46);
            this.selectedSizeListBox.Name = "selectedSizeListBox";
            this.selectedSizeListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedSizeListBox.TabIndex = 25;
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(6, 30);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(125, 13);
            this.label11.TabIndex = 20;
            this.label11.Text = "Available Measurements:";
            //
            // label12
            //
            this.label12.AutoSize = true;
```

```
            this.label12.Location = new System.Drawing.Point(172, 27);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(124, 13);
            this.label12.TabIndex = 24;
            this.label12.Text = "Selected Measurements:";
            //
            // sizeRemoveButton
            //
            this.sizeRemoveButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.sizeRemoveButton.Location = new System.Drawing.Point(130, 109);
            this.sizeRemoveButton.Name = "sizeRemoveButton";
            this.sizeRemoveButton.Size = new System.Drawing.Size(39, 38);
            this.sizeRemoveButton.TabIndex = 23;
            this.sizeRemoveButton.Text = "<<";
            this.sizeRemoveButton.UseVisualStyleBackColor = true;
            this.sizeRemoveButton.Click += new System.EventHandler(this.sizeRemoveButton_Click);
            //
            // label13
            //
            this.label13.AutoSize = true;
            this.label13.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label13.Location = new System.Drawing.Point(6, 7);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(200, 17);
            this.label13.TabIndex = 34;
            this.label13.Text = "Product Size and Stability:";
            //
            // performanceGroupBox
            //
            this.performanceGroupBox.Controls.Add(this.performanceListBox);
            this.performanceGroupBox.Controls.Add(this.performanceSelectButton);
            this.performanceGroupBox.Controls.Add(this.selectedPerformanceListBox);
            this.performanceGroupBox.Controls.Add(this.label14);
            this.performanceGroupBox.Controls.Add(this.label15);
            this.performanceGroupBox.Controls.Add(this.performanceRemoveButton);
            this.performanceGroupBox.Controls.Add(this.label16);
            this.performanceGroupBox.Location = new System.Drawing.Point(656, 12);
            this.performanceGroupBox.Name = "performanceGroupBox";
            this.performanceGroupBox.Size = new System.Drawing.Size(311, 162);
            this.performanceGroupBox.TabIndex = 48;
            this.performanceGroupBox.TabStop = false;
            //
            // performanceListBox
            //
            this.performanceListBox.FormattingEnabled = true;
            this.performanceListBox.Location = new System.Drawing.Point(6, 46);
            this.performanceListBox.Name = "performanceListBox";
            this.performanceListBox.Size = new System.Drawing.Size(118, 108);
            this.performanceListBox.TabIndex = 21;
            //
            // performanceSelectButton
```

```
            //
            this.performanceSelectButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.performanceSelectButton.Location = new System.Drawing.Point(130, 55);
            this.performanceSelectButton.Name = "performanceSelectButton";
            this.performanceSelectButton.Size = new System.Drawing.Size(39, 38);
            this.performanceSelectButton.TabIndex = 22;
            this.performanceSelectButton.Text = ">>";
            this.performanceSelectButton.UseVisualStyleBackColor = true;
            this.performanceSelectButton.Click += new
System.EventHandler(this.performanceSelectButton_Click);
            //
            // selectedPerformanceListBox
            //
            this.selectedPerformanceListBox.FormattingEnabled = true;
            this.selectedPerformanceListBox.Location = new System.Drawing.Point(175, 46);
            this.selectedPerformanceListBox.Name = "selectedPerformanceListBox";
            this.selectedPerformanceListBox.Size = new System.Drawing.Size(118, 108);
            this.selectedPerformanceListBox.TabIndex = 25;
            //
            // label14
            //
            this.label14.AutoSize = true;
            this.label14.Location = new System.Drawing.Point(6, 30);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(125, 13);
            this.label14.TabIndex = 20;
            this.label14.Text = "Available Measurements:";
            //
            // label15
            //
            this.label15.AutoSize = true;
            this.label15.Location = new System.Drawing.Point(172, 27);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(124, 13);
            this.label15.TabIndex = 24;
            this.label15.Text = "Selected Measurements:";
            //
            // performanceRemoveButton
            //
            this.performanceRemoveButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.performanceRemoveButton.Location = new System.Drawing.Point(130, 109);
            this.performanceRemoveButton.Name = "performanceRemoveButton";
            this.performanceRemoveButton.Size = new System.Drawing.Size(39, 38);
            this.performanceRemoveButton.TabIndex = 23;
            this.performanceRemoveButton.Text = "<<";
            this.performanceRemoveButton.UseVisualStyleBackColor = true;
            this.performanceRemoveButton.Click += new
System.EventHandler(this.performanceRemoveButton_Click);
            //
            // label16
            //
```

```
            this.label16.AutoSize = true;
            this.label16.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label16.Location = new System.Drawing.Point(6, 7);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(168, 17);
            this.label16.TabIndex = 34;
            this.label16.Text = "Process Performance:";
            //
            // MeasurementForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(977, 358);
            this.Controls.Add(this.performanceGroupBox);
            this.Controls.Add(this.qualityGroupBox);
            this.Controls.Add(this.sizeGroupBox);
            this.Controls.Add(this.resourcesGroupBox);
            this.Controls.Add(this.ScheduleGroupBox);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.selectedMeasurementListBox);
            this.Controls.Add(this.submitButton);
            this.Name = "MeasurementForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "MeasurementForm";
            this.ScheduleGroupBox.ResumeLayout(false);
            this.ScheduleGroupBox.PerformLayout();
            this.resourcesGroupBox.ResumeLayout(false);
            this.resourcesGroupBox.PerformLayout();
            this.qualityGroupBox.ResumeLayout(false);
            this.qualityGroupBox.PerformLayout();
            this.sizeGroupBox.ResumeLayout(false);
            this.sizeGroupBox.PerformLayout();
            this.performanceGroupBox.ResumeLayout(false);
            this.performanceGroupBox.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Button scheduleRemoveButton;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ListBox selectedScheduleListBox;
        private System.Windows.Forms.Button scheduleSelectButton;
        private System.Windows.Forms.ListBox scheduleListBox;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.ListBox selectedMeasurementListBox;
        private System.Windows.Forms.GroupBox ScheduleGroupBox;
```

```
        private System.Windows.Forms.GroupBox resourcesGroupBox;
        private System.Windows.Forms.ListBox resourcesListBox;
        private System.Windows.Forms.Button resourcesSelectButton;
        private System.Windows.Forms.ListBox selectedResourcesListBox;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Button resourcesRemoveButton;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.GroupBox qualityGroupBox;
        private System.Windows.Forms.ListBox qualityListBox;
        private System.Windows.Forms.Button qualitySelectButton;
        private System.Windows.Forms.ListBox selectedQualityListBox;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Button qualityRemoveButton;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.GroupBox sizeGroupBox;
        private System.Windows.Forms.ListBox sizeListBox;
        private System.Windows.Forms.Button sizeSelectButton;
        private System.Windows.Forms.ListBox selectedSizeListBox;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.Label label12;
        private System.Windows.Forms.Button sizeRemoveButton;
        private System.Windows.Forms.Label label13;
        private System.Windows.Forms.GroupBox performanceGroupBox;
        private System.Windows.Forms.ListBox performanceListBox;
        private System.Windows.Forms.Button performanceSelectButton;
        private System.Windows.Forms.ListBox selectedPerformanceListBox;
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.Button performanceRemoveButton;
        private System.Windows.Forms.Label label16;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace SDPTool
{
   public partial class MeasurementForm : Form
   {
      static int MAX_MEASUREMENTS = 10;
      string[] measures = new string[MAX_MEASUREMENTS];
      int count;

      public string[] Measures
      {
         get  //get accessor method
         {
            return measures;
         }
      }

      public int Count
      {
         get  //get accessor method
         {
            return count;
         }
      }

      public MeasurementForm()
      {
         InitializeComponent();
         AddMeasurement();
      }

      private void AddMeasurement()
      {
         scheduleListBox.Items.Add("ComponentStatus");
         scheduleListBox.Items.Add("ProjectVelocity"); //(userstory/time)

         resourcesListBox.Items.Add("PersonnelEffort"); //time
         resourcesListBox.Items.Add("EarnedValue");

         sizeListBox.Items.Add("ComponentSize");
         sizeListBox.Items.Add("LinesOfCode");

         qualityListBox.Items.Add("Defects");

         performanceListBox.Items.Add("Productivity"); //(LOC/time)
      }
```

```
        private void submitButton_Click(object sender, EventArgs e)
        {
            count = 0;
            foreach (string item in selectedScheduleListBox.Items)
            {
                measures[count++] = item.Trim();
            }
            foreach (string item in selectedResourcesListBox.Items)
            {
                measures[count++] = item.Trim();
            }
            foreach (string item in selectedPerformanceListBox.Items)
            {
                measures[count++] = item.Trim();
            }
            foreach (string item in selectedSizeListBox.Items)
            {
                measures[count++] = item.Trim();
            }
            foreach (string item in selectedQualityListBox.Items)
            {
                measures[count++] = item.Trim();
            }
            Close();
        }

        private void scheduleSelectButton_Click(object sender, EventArgs e)
        {
            string selectedText = (string)scheduleListBox.SelectedItem;
            if (selectedText != null)
            {
                selectedScheduleListBox.Items.Add(selectedText);
                scheduleListBox.Items.Remove(scheduleListBox.SelectedItem);
                scheduleRemoveButton.Enabled = true;
            }
            else
            {
                MessageBox.Show("Please Select an Measurement to add", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void scheduleRemoveButton_Click(object sender, EventArgs e)
        {
            string selectedItem = (string)selectedScheduleListBox.SelectedItem;
            if (selectedItem != null)
            {
                selectedScheduleListBox.Items.Remove(selectedItem);
                scheduleListBox.Items.Add(selectedItem);
                if (selectedScheduleListBox.Items.Count == 0)
                    scheduleRemoveButton.Enabled = false;
            }
```

```
    }

    private void resourcesSelectButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)resourcesListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedResourcesListBox.Items.Add(selectedText);
            resourcesListBox.Items.Remove(resourcesListBox.SelectedItem);
            resourcesRemoveButton.Enabled = true;
        }
        else
        {
            MessageBox.Show("Please Select an Measurement to add", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void resourcesRemoveButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)selectedResourcesListBox.SelectedItem;
        if (selectedItem != null)
        {
            selectedResourcesListBox.Items.Remove(selectedItem);
            resourcesListBox.Items.Add(selectedItem);
            if (selectedResourcesListBox.Items.Count == 0)
                resourcesRemoveButton.Enabled = false;
        }
    }


    private void performanceSelectButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)performanceListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedPerformanceListBox.Items.Add(selectedText);
            performanceListBox.Items.Remove(performanceListBox.SelectedItem);
            performanceRemoveButton.Enabled = true;
        }
        else
        {
            MessageBox.Show("Please Select an Measurement to add", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void performanceRemoveButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)selectedPerformanceListBox.SelectedItem;
        if (selectedItem != null)
        {
            selectedPerformanceListBox.Items.Remove(selectedItem);
```

```
            performanceListBox.Items.Add(selectedItem);
            if (selectedPerformanceListBox.Items.Count == 0)
                performanceRemoveButton.Enabled = false;
        }
    }

    private void sizeSelectButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)sizeListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedSizeListBox.Items.Add(selectedText);
            sizeListBox.Items.Remove(sizeListBox.SelectedItem);
            sizeRemoveButton.Enabled = true;
        }
        else
        {
            MessageBox.Show("Please Select an Measurement to add", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void sizeRemoveButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)selectedSizeListBox.SelectedItem;
        if (selectedItem != null)
        {
            selectedSizeListBox.Items.Remove(selectedItem);
            sizeListBox.Items.Add(selectedItem);
            if (selectedSizeListBox.Items.Count == 0)
                sizeRemoveButton.Enabled = false;
        }
    }

    private void qualitySelectButton_Click(object sender, EventArgs e)
    {
        string selectedText = (string)qualityListBox.SelectedItem;
        if (selectedText != null)
        {
            selectedQualityListBox.Items.Add(selectedText);
            qualityListBox.Items.Remove(qualityListBox.SelectedItem);
            qualityRemoveButton.Enabled = true;
        }
        else
        {
            MessageBox.Show("Please Select an Measurement to add", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void qualityRemoveButton_Click(object sender, EventArgs e)
    {
        string selectedItem = (string)selectedQualityListBox.SelectedItem;
```

```
            if (selectedItem != null)
            {
                selectedQualityListBox.Items.Remove(selectedItem);
                qualityListBox.Items.Add(selectedItem);
                if (selectedQualityListBox.Items.Count == 0)
                    qualityRemoveButton.Enabled = false;
            }
        }
    }
}
```

```
namespace SDPTool
{
   partial class InitializeForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.submitButton = new System.Windows.Forms.Button();
         this.clearButton = new System.Windows.Forms.Button();
         this.label3 = new System.Windows.Forms.Label();
         this.label6 = new System.Windows.Forms.Label();
         this.label1 = new System.Windows.Forms.Label();
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.label2 = new System.Windows.Forms.Label();
         this.textBox5 = new System.Windows.Forms.TextBox();
         this.textBox3 = new System.Windows.Forms.TextBox();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.yesRadioButton = new System.Windows.Forms.RadioButton();
         this.label5 = new System.Windows.Forms.Label();
         this.noRadioButton = new System.Windows.Forms.RadioButton();
         this.textBox4 = new System.Windows.Forms.TextBox();
         this.label4 = new System.Windows.Forms.Label();
         this.exitButton = new System.Windows.Forms.Button();
         this.label7 = new System.Windows.Forms.Label();
         this.employeeYesRadioButton = new System.Windows.Forms.RadioButton();
         this.employeeNoRadioButton = new System.Windows.Forms.RadioButton();
         this.employeeListGroupBox = new System.Windows.Forms.GroupBox();
         this.pauseGroupBox = new System.Windows.Forms.GroupBox();
         this.employeeListGroupBox.SuspendLayout();
```

```
this.pauseGroupBox.SuspendLayout();
this.SuspendLayout();
//
// submitButton
//
this.submitButton.Location = new System.Drawing.Point(107, 208);
this.submitButton.Name = "submitButton";
this.submitButton.Size = new System.Drawing.Size(75, 23);
this.submitButton.TabIndex = 12;
this.submitButton.Text = "Submit";
this.submitButton.UseVisualStyleBackColor = true;
this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
//
// clearButton
//
this.clearButton.Location = new System.Drawing.Point(188, 208);
this.clearButton.Name = "clearButton";
this.clearButton.Size = new System.Drawing.Size(75, 23);
this.clearButton.TabIndex = 13;
this.clearButton.Text = "Clear";
this.clearButton.UseVisualStyleBackColor = true;
this.clearButton.Click += new System.EventHandler(this.clearButton_Click);
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(203, 53);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(104, 13);
this.label3.TabIndex = 4;
this.label3.Text = "Project Size (in LOC)";
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(9, 169);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(109, 13);
this.label6.TabIndex = 10;
this.label6.Text = "Time Interval (in mins)";
this.label6.Visible = false;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(9, 20);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(110, 13);
this.label1.TabIndex = 0;
this.label1.Text = "Number of Employees";
//
// textBox1
//
```

```
            this.textBox1.Location = new System.Drawing.Point(124, 17);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(47, 20);
            this.textBox1.TabIndex = 1;
            this.textBox1.TextChanged += new System.EventHandler(this.textBox1_TextChanged);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(216, 20);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(91, 13);
            this.label2.TabIndex = 15;
            this.label2.Text = "Number of Teams";
            //
            // textBox5
            //
            this.textBox5.Location = new System.Drawing.Point(124, 166);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(47, 20);
            this.textBox5.TabIndex = 11;
            this.textBox5.Visible = false;
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(313, 50);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(47, 20);
            this.textBox3.TabIndex = 5;
            this.textBox3.Text = "NA";
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(313, 17);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(47, 20);
            this.textBox2.TabIndex = 16;
            //
            // yesRadioButton
            //
            this.yesRadioButton.AutoSize = true;
            this.yesRadioButton.Location = new System.Drawing.Point(5, 8);
            this.yesRadioButton.Name = "yesRadioButton";
            this.yesRadioButton.Size = new System.Drawing.Size(43, 17);
            this.yesRadioButton.TabIndex = 0;
            this.yesRadioButton.Text = "Yes";
            this.yesRadioButton.UseVisualStyleBackColor = true;
            this.yesRadioButton.CheckedChanged += new
System.EventHandler(this.yesRadioButton_CheckedChanged);
            //
            // label5
            //
            this.label5.AutoSize = true;
```

```
            this.label5.Location = new System.Drawing.Point(12, 121);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(244, 13);
            this.label5.TabIndex = 8;
            this.label5.Text = "Want simulation to pause after every time interval?";
            //
            // noRadioButton
            //
            this.noRadioButton.AutoSize = true;
            this.noRadioButton.Checked = true;
            this.noRadioButton.Location = new System.Drawing.Point(54, 8);
            this.noRadioButton.Name = "noRadioButton";
            this.noRadioButton.Size = new System.Drawing.Size(39, 17);
            this.noRadioButton.TabIndex = 1;
            this.noRadioButton.TabStop = true;
            this.noRadioButton.Text = "No";
            this.noRadioButton.UseVisualStyleBackColor = true;
            this.noRadioButton.CheckedChanged += new
System.EventHandler(this.noRadioButton_CheckedChanged);
            //
            // textBox4
            //
            this.textBox4.Location = new System.Drawing.Point(313, 80);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(47, 20);
            this.textBox4.TabIndex = 7;
            this.textBox4.WordWrap = false;
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(191, 83);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(116, 13);
            this.label4.TabIndex = 6;
            this.label4.Text = "Number of User Stories";
            //
            // exitButton
            //
            this.exitButton.Location = new System.Drawing.Point(269, 208);
            this.exitButton.Name = "exitButton";
            this.exitButton.Size = new System.Drawing.Size(75, 23);
            this.exitButton.TabIndex = 14;
            this.exitButton.Text = "Exit";
            this.exitButton.UseVisualStyleBackColor = true;
            this.exitButton.Click += new System.EventHandler(this.exitButton_Click);
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(9, 53);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(154, 13);
```

```
            this.label7.TabIndex = 2;
            this.label7.Text = "Employee List already created?";
            //
            // employeeYesRadioButton
            //
            this.employeeYesRadioButton.AutoSize = true;
            this.employeeYesRadioButton.Location = new System.Drawing.Point(5, 7);
            this.employeeYesRadioButton.Name = "employeeYesRadioButton";
            this.employeeYesRadioButton.Size = new System.Drawing.Size(43, 17);
            this.employeeYesRadioButton.TabIndex = 0;
            this.employeeYesRadioButton.Text = "Yes";
            this.employeeYesRadioButton.UseVisualStyleBackColor = true;
            this.employeeYesRadioButton.CheckedChanged += new
System.EventHandler(this.employeeYesRadioButton_CheckedChanged);
            //
            // employeeNoRadioButton
            //
            this.employeeNoRadioButton.AutoSize = true;
            this.employeeNoRadioButton.Checked = true;
            this.employeeNoRadioButton.Location = new System.Drawing.Point(54, 7);
            this.employeeNoRadioButton.Name = "employeeNoRadioButton";
            this.employeeNoRadioButton.Size = new System.Drawing.Size(39, 17);
            this.employeeNoRadioButton.TabIndex = 1;
            this.employeeNoRadioButton.TabStop = true;
            this.employeeNoRadioButton.Text = "No";
            this.employeeNoRadioButton.UseVisualStyleBackColor = true;
            this.employeeNoRadioButton.CheckedChanged += new
System.EventHandler(this.employeeNoRadioButton_CheckedChanged);
            //
            // employeeListGroupBox
            //
            this.employeeListGroupBox.Controls.Add(this.employeeNoRadioButton);
            this.employeeListGroupBox.Controls.Add(this.employeeYesRadioButton);
            this.employeeListGroupBox.Location = new System.Drawing.Point(47, 68);
            this.employeeListGroupBox.Margin = new System.Windows.Forms.Padding(2);
            this.employeeListGroupBox.Name = "employeeListGroupBox";
            this.employeeListGroupBox.Padding = new System.Windows.Forms.Padding(2);
            this.employeeListGroupBox.Size = new System.Drawing.Size(103, 28);
            this.employeeListGroupBox.TabIndex = 3;
            this.employeeListGroupBox.TabStop = false;
            //
            // pauseGroupBox
            //
            this.pauseGroupBox.Controls.Add(this.yesRadioButton);
            this.pauseGroupBox.Controls.Add(this.noRadioButton);
            this.pauseGroupBox.Location = new System.Drawing.Point(124, 132);
            this.pauseGroupBox.Margin = new System.Windows.Forms.Padding(2);
            this.pauseGroupBox.Name = "pauseGroupBox";
            this.pauseGroupBox.Padding = new System.Windows.Forms.Padding(2);
            this.pauseGroupBox.Size = new System.Drawing.Size(104, 29);
            this.pauseGroupBox.TabIndex = 9;
            this.pauseGroupBox.TabStop = false;
            //
```

```
    // InitializeForm
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(381, 243);
    this.Controls.Add(this.pauseGroupBox);
    this.Controls.Add(this.employeeListGroupBox);
    this.Controls.Add(this.label7);
    this.Controls.Add(this.exitButton);
    this.Controls.Add(this.textBox4);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.label5);
    this.Controls.Add(this.textBox2);
    this.Controls.Add(this.textBox3);
    this.Controls.Add(this.textBox5);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.label6);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.textBox1);
    this.Controls.Add(this.clearButton);
    this.Controls.Add(this.submitButton);
    this.Name = "InitializeForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
    this.Text = "Simulation Paramenters";
    this.employeeListGroupBox.ResumeLayout(false);
    this.employeeListGroupBox.PerformLayout();
    this.pauseGroupBox.ResumeLayout(false);
    this.pauseGroupBox.PerformLayout();
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button submitButton;
private System.Windows.Forms.Button clearButton;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.TextBox textBox5;
private System.Windows.Forms.TextBox textBox3;
private System.Windows.Forms.TextBox textBox2;
private System.Windows.Forms.RadioButton yesRadioButton;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.RadioButton noRadioButton;
private System.Windows.Forms.TextBox textBox4;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Button exitButton;
private System.Windows.Forms.Label label7;
```

```
        private System.Windows.Forms.RadioButton employeeYesRadioButton;
        private System.Windows.Forms.RadioButton employeeNoRadioButton;
        private System.Windows.Forms.GroupBox employeeListGroupBox;
        private System.Windows.Forms.GroupBox pauseGroupBox;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{
   public partial class InitializeForm : Form
   {
      bool pause = false;
      bool employeeListCreated = false;
      string fileName;

      public string FileName
      {
         get  //get accessor method
         {
            return fileName;
         }
         set   //set accessor method
         {
            fileName = value;
         }
      }

      public InitializeForm()
      {
         InitializeComponent();
      }

      private void submitButton_Click(object sender, EventArgs e)
      {
         try
         {
            FileStream file;
            StreamWriter sw;
            file = new FileStream(FileName, FileMode.Create, FileAccess.Write);
            sw = new StreamWriter(file);
            sw.Write(textBox1.Text); sw.Write(", ");
            sw.Write(textBox2.Text); sw.Write(", ");
            if ((textBox3.Text.Trim().Equals("NA")) || (textBox3.Text.Trim().Equals(null)))
            {
               sw.Write(-1); sw.Write(", ");
            }
            else
            {
               sw.Write(textBox3.Text); sw.Write(", ");
            }
            sw.Write(textBox4.Text); sw.Write(", ");
```

```
            if (pause)
            {
               sw.Write("true"); sw.Write(", ");
               sw.Write(textBox5.Text); sw.Write(", ");
            }
            else
            {
               sw.Write("false"); sw.Write(", ");
               sw.Write("0"); sw.Write(", ");
            }

            if (employeeListCreated.Equals(false))
            {
               AgentForm agentForm1 = new AgentForm();
               agentForm1.ShowDialog();
               sw.Write("false");
            }
            else
            {
               sw.Write("true");
            }
            sw.WriteLine();
            sw.Close();
            file.Close();
            this.Close();

         }
         catch
         {
            MessageBox.Show("The values are not valid. Please check the values again", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
      }

      private void clearButton_Click(object sender, EventArgs e)
      {
         textBox1.Clear();
         textBox2.Clear();
         textBox3.Clear();
         textBox3.Text = "NA";
         textBox4.Clear();
         textBox5.Clear();
      }

      private void yesRadioButton_CheckedChanged(object sender, EventArgs e)
      {
         pause = true;
         label6.Visible = true;
         textBox5.Visible = true;
      }

      private void noRadioButton_CheckedChanged(object sender, EventArgs e)
      {
```

```
            pause = false;
            label6.Visible = false;
            textBox5.Visible = false;
        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            this.Close();
            Environment.Exit(0);
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            if (textBox1.TextLength > 0)
            {
                if (Convert.ToInt32(textBox1.Text) > 0)
                {
                    textBox2.Text = Convert.ToString(Convert.ToInt32(textBox1.Text) / 2);
                }
            }
            else
                textBox2.Text = null;
        }

        private void employeeYesRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            employeeListCreated = true;
        }

        private void employeeNoRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            employeeListCreated = false;
        }
    }
}
```

```
using System;
using System.IO;

namespace SDPTool
{
   class File
   {
      FileStream file;
      StreamReader sr;
      StreamWriter sw;
      string fileName;

      public File()
      {
         fileName = "unnamed.txt";
      }

      public File(string name)
      {
         fileName = name;
      }

      public string Read()
      {
         file = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
         sr = new StreamReader(file);
         string s = sr.ReadToEnd();
         Console.WriteLine(s);
         sr.Close();
         file.Close();
         return s;
      }

      public string ReadLine()
      {
         file = new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Read);
         sr = new StreamReader(file);
         string s = sr.ReadLine();
         Console.WriteLine(s);
         //sr.Close();
         //file.Close();
         return s;
      }

      public void Write(string s)
      {
         file = new FileStream(fileName, FileMode.Append, FileAccess.Write);
         sw = new StreamWriter(file);
         sw.Write(s);
         sw.Close();
         file.Close();
      }
```

```
        public void WriteLine(string s)
        {
            file = new FileStream(fileName, FileMode.Append, FileAccess.Write);
            sw = new StreamWriter(file);
            sw.WriteLine(s);
            sw.Close();
            file.Close();
        }
    }
}
```

```
namespace SDPTool
{
   partial class AgentForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.label1 = new System.Windows.Forms.Label();
         this.label2 = new System.Windows.Forms.Label();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.label3 = new System.Windows.Forms.Label();
         this.textBox3 = new System.Windows.Forms.TextBox();
         this.label4 = new System.Windows.Forms.Label();
         this.label5 = new System.Windows.Forms.Label();
         this.textBox5 = new System.Windows.Forms.TextBox();
         this.label6 = new System.Windows.Forms.Label();
         this.textBox6 = new System.Windows.Forms.TextBox();
         this.label7 = new System.Windows.Forms.Label();
         this.textBox7 = new System.Windows.Forms.TextBox();
         this.label8 = new System.Windows.Forms.Label();
         this.textBox8 = new System.Windows.Forms.TextBox();
         this.label9 = new System.Windows.Forms.Label();
         this.textBox9 = new System.Windows.Forms.TextBox();
         this.label10 = new System.Windows.Forms.Label();
         this.textBox10 = new System.Windows.Forms.TextBox();
         this.label11 = new System.Windows.Forms.Label();
         this.textBox11 = new System.Windows.Forms.TextBox();
         this.label12 = new System.Windows.Forms.Label();
```

```
this.textBox12 = new System.Windows.Forms.TextBox();
this.label13 = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.textBox13 = new System.Windows.Forms.TextBox();
this.label15 = new System.Windows.Forms.Label();
this.textBox14 = new System.Windows.Forms.TextBox();
this.label16 = new System.Windows.Forms.Label();
this.textBox15 = new System.Windows.Forms.TextBox();
this.label17 = new System.Windows.Forms.Label();
this.textBox16 = new System.Windows.Forms.TextBox();
this.label18 = new System.Windows.Forms.Label();
this.label19 = new System.Windows.Forms.Label();
this.textBox17 = new System.Windows.Forms.TextBox();
this.label22 = new System.Windows.Forms.Label();
this.textBox20 = new System.Windows.Forms.TextBox();
this.label23 = new System.Windows.Forms.Label();
this.textBox21 = new System.Windows.Forms.TextBox();
this.label24 = new System.Windows.Forms.Label();
this.textBox22 = new System.Windows.Forms.TextBox();
this.label25 = new System.Windows.Forms.Label();
this.textBox23 = new System.Windows.Forms.TextBox();
this.label26 = new System.Windows.Forms.Label();
this.textBox24 = new System.Windows.Forms.TextBox();
this.label27 = new System.Windows.Forms.Label();
this.label28 = new System.Windows.Forms.Label();
this.textBox25 = new System.Windows.Forms.TextBox();
this.label31 = new System.Windows.Forms.Label();
this.textBox28 = new System.Windows.Forms.TextBox();
this.label32 = new System.Windows.Forms.Label();
this.textBox29 = new System.Windows.Forms.TextBox();
this.label33 = new System.Windows.Forms.Label();
this.textBox30 = new System.Windows.Forms.TextBox();
this.label34 = new System.Windows.Forms.Label();
this.textBox31 = new System.Windows.Forms.TextBox();
this.label35 = new System.Windows.Forms.Label();
this.textBox32 = new System.Windows.Forms.TextBox();
this.label36 = new System.Windows.Forms.Label();
this.label37 = new System.Windows.Forms.Label();
this.textBox33 = new System.Windows.Forms.TextBox();
this.label40 = new System.Windows.Forms.Label();
this.textBox27 = new System.Windows.Forms.TextBox();
this.label41 = new System.Windows.Forms.Label();
this.textBox34 = new System.Windows.Forms.TextBox();
this.label42 = new System.Windows.Forms.Label();
this.textBox26 = new System.Windows.Forms.TextBox();
this.label20 = new System.Windows.Forms.Label();
this.textBox18 = new System.Windows.Forms.TextBox();
this.label21 = new System.Windows.Forms.Label();
this.textBox19 = new System.Windows.Forms.TextBox();
this.submitButton = new System.Windows.Forms.Button();
this.clearAllButton = new System.Windows.Forms.Button();
this.closeButton = new System.Windows.Forms.Button();
this.label29 = new System.Windows.Forms.Label();
```

```
this.textBox4 = new System.Windows.Forms.TextBox();
this.label30 = new System.Windows.Forms.Label();
this.textBox35 = new System.Windows.Forms.TextBox();
this.SuspendLayout();
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(110, 24);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(132, 20);
this.textBox1.TabIndex = 1;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(12, 31);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(67, 13);
this.label1.TabIndex = 0;
this.label1.Text = "Employee ID";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(262, 31);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(35, 13);
this.label2.TabIndex = 2;
this.label2.Text = "Name";
//
// textBox2
//
this.textBox2.Location = new System.Drawing.Point(356, 23);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(132, 20);
this.textBox2.TabIndex = 3;
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(17, 96);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(83, 13);
this.label3.TabIndex = 5;
this.label3.Text = "LOC Developed";
//
// textBox3
//
this.textBox3.Location = new System.Drawing.Point(110, 89);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(132, 20);
this.textBox3.TabIndex = 6;
//
```

```
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.Location = new System.Drawing.Point(16, 613);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(36, 13);
            this.label4.TabIndex = 60;
            this.label4.Text = "Total";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(21, 287);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(32, 13);
            this.label5.TabIndex = 22;
            this.label5.Text = "Code";
            //
            // textBox5
            //
            this.textBox5.Location = new System.Drawing.Point(356, 115);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(132, 20);
            this.textBox5.TabIndex = 10;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(21, 205);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(48, 13);
            this.label6.TabIndex = 16;
            this.label6.Text = "Planning";
            //
            // textBox6
            //
            this.textBox6.Location = new System.Drawing.Point(343, 361);
            this.textBox6.Name = "textBox6";
            this.textBox6.Size = new System.Drawing.Size(132, 20);
            this.textBox6.TabIndex = 46;
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(278, 364);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(36, 13);
            this.label7.TabIndex = 45;
            this.label7.Text = "Total";
```

```
//
// textBox7
//
this.textBox7.Location = new System.Drawing.Point(110, 141);
this.textBox7.Name = "textBox7";
this.textBox7.Size = new System.Drawing.Size(132, 20);
this.textBox7.TabIndex = 12;
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(262, 119);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(88, 13);
this.label8.TabIndex = 9;
this.label8.Text = "Upper LOC/Hour";
//
// textBox8
//
this.textBox8.Location = new System.Drawing.Point(356, 141);
this.textBox8.Name = "textBox8";
this.textBox8.Size = new System.Drawing.Size(132, 20);
this.textBox8.TabIndex = 14;
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(21, 340);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(44, 13);
this.label9.TabIndex = 26;
this.label9.Text = "Compile";
//
// textBox9
//
this.textBox9.Location = new System.Drawing.Point(110, 205);
this.textBox9.Name = "textBox9";
this.textBox9.Size = new System.Drawing.Size(132, 20);
this.textBox9.TabIndex = 17;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(21, 261);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(79, 13);
this.label10.TabIndex = 20;
this.label10.Text = "Design Review";
//
// textBox10
//
this.textBox10.Location = new System.Drawing.Point(110, 231);
this.textBox10.Name = "textBox10";
```

```
            this.textBox10.Size = new System.Drawing.Size(132, 20);
            this.textBox10.TabIndex = 19;
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(21, 313);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(71, 13);
            this.label11.TabIndex = 24;
            this.label11.Text = "Code Review";
            //
            // textBox11
            //
            this.textBox11.Location = new System.Drawing.Point(110, 258);
            this.textBox11.Name = "textBox11";
            this.textBox11.Size = new System.Drawing.Size(132, 20);
            this.textBox11.TabIndex = 21;
            //
            // label12
            //
            this.label12.AutoSize = true;
            this.label12.Location = new System.Drawing.Point(21, 234);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(40, 13);
            this.label12.TabIndex = 18;
            this.label12.Text = "Design";
            //
            // textBox12
            //
            this.textBox12.Location = new System.Drawing.Point(110, 284);
            this.textBox12.Name = "textBox12";
            this.textBox12.Size = new System.Drawing.Size(132, 20);
            this.textBox12.TabIndex = 23;
            //
            // label13
            //
            this.label13.AutoSize = true;
            this.label13.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label13.Location = new System.Drawing.Point(70, 189);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(129, 13);
            this.label13.TabIndex = 15;
            this.label13.Text = "Time In Phases (in %)";
            //
            // label14
            //
            this.label14.AutoSize = true;
            this.label14.Location = new System.Drawing.Point(23, 393);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(108, 13);
            this.label14.TabIndex = 30;
```

```
this.label14.Text = "Postmortem/Refactor";
//
// textBox13
//
this.textBox13.Location = new System.Drawing.Point(110, 310);
this.textBox13.Name = "textBox13";
this.textBox13.Size = new System.Drawing.Size(132, 20);
this.textBox13.TabIndex = 25;
//
// label15
//
this.label15.AutoSize = true;
this.label15.Location = new System.Drawing.Point(23, 366);
this.label15.Name = "label15";
this.label15.Size = new System.Drawing.Size(28, 13);
this.label15.TabIndex = 28;
this.label15.Text = "Test";
//
// textBox14
//
this.textBox14.Location = new System.Drawing.Point(110, 336);
this.textBox14.Name = "textBox14";
this.textBox14.Size = new System.Drawing.Size(132, 20);
this.textBox14.TabIndex = 27;
//
// label16
//
this.label16.AutoSize = true;
this.label16.Location = new System.Drawing.Point(277, 343);
this.label16.Name = "label16";
this.label16.Size = new System.Drawing.Size(62, 13);
this.label16.TabIndex = 43;
this.label16.Text = "Postmortem";
//
// textBox15
//
this.textBox15.Location = new System.Drawing.Point(110, 362);
this.textBox15.Name = "textBox15";
this.textBox15.Size = new System.Drawing.Size(132, 20);
this.textBox15.TabIndex = 29;
//
// label17
//
this.label17.AutoSize = true;
this.label17.Location = new System.Drawing.Point(277, 316);
this.label17.Name = "label17";
this.label17.Size = new System.Drawing.Size(28, 13);
this.label17.TabIndex = 41;
this.label17.Text = "Test";
//
// textBox16
//
this.textBox16.Location = new System.Drawing.Point(137, 388);
```

```
            this.textBox16.Name = "textBox16";
            this.textBox16.Size = new System.Drawing.Size(126, 20);
            this.textBox16.TabIndex = 31;
            //
            // label18
            //
            this.label18.AutoSize = true;
            this.label18.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label18.Location = new System.Drawing.Point(310, 189);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(101, 13);
            this.label18.TabIndex = 32;
            this.label18.Text = "Defects Injected";
            //
            // label19
            //
            this.label19.AutoSize = true;
            this.label19.Location = new System.Drawing.Point(275, 290);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(44, 13);
            this.label19.TabIndex = 39;
            this.label19.Text = "Compile";
            //
            // textBox17
            //
            this.textBox17.Location = new System.Drawing.Point(343, 205);
            this.textBox17.Name = "textBox17";
            this.textBox17.Size = new System.Drawing.Size(132, 20);
            this.textBox17.TabIndex = 34;
            this.textBox17.TextChanged += new System.EventHandler(this.textBox17_TextChanged);
            //
            // label22
            //
            this.label22.AutoSize = true;
            this.label22.Location = new System.Drawing.Point(275, 237);
            this.label22.Name = "label22";
            this.label22.Size = new System.Drawing.Size(40, 13);
            this.label22.TabIndex = 35;
            this.label22.Text = "Design";
            //
            // textBox20
            //
            this.textBox20.Location = new System.Drawing.Point(343, 283);
            this.textBox20.Name = "textBox20";
            this.textBox20.Size = new System.Drawing.Size(132, 20);
            this.textBox20.TabIndex = 40;
            this.textBox20.TextChanged += new System.EventHandler(this.textBox20_TextChanged);
            //
            // label23
            //
            this.label23.AutoSize = true;
            this.label23.Location = new System.Drawing.Point(275, 264);
```

```
this.label23.Name = "label23";
this.label23.Size = new System.Drawing.Size(32, 13);
this.label23.TabIndex = 37;
this.label23.Text = "Code";
//
// textBox21
//
this.textBox21.Location = new System.Drawing.Point(343, 309);
this.textBox21.Name = "textBox21";
this.textBox21.Size = new System.Drawing.Size(132, 20);
this.textBox21.TabIndex = 42;
this.textBox21.TextChanged += new System.EventHandler(this.textBox21_TextChanged);
//
// label24
//
this.label24.AutoSize = true;
this.label24.Location = new System.Drawing.Point(275, 208);
this.label24.Name = "label24";
this.label24.Size = new System.Drawing.Size(48, 13);
this.label24.TabIndex = 33;
this.label24.Text = "Planning";
//
// textBox22
//
this.textBox22.Location = new System.Drawing.Point(343, 335);
this.textBox22.Name = "textBox22";
this.textBox22.Size = new System.Drawing.Size(132, 20);
this.textBox22.TabIndex = 44;
this.textBox22.TextChanged += new System.EventHandler(this.textBox22_TextChanged);
//
// label25
//
this.label25.AutoSize = true;
this.label25.Location = new System.Drawing.Point(278, 588);
this.label25.Name = "label25";
this.label25.Size = new System.Drawing.Size(62, 13);
this.label25.TabIndex = 73;
this.label25.Text = "Postmortem";
//
// textBox23
//
this.textBox23.Location = new System.Drawing.Point(89, 450);
this.textBox23.Name = "textBox23";
this.textBox23.Size = new System.Drawing.Size(132, 20);
this.textBox23.TabIndex = 49;
this.textBox23.TextChanged += new System.EventHandler(this.textBox23_TextChanged);
//
// label26
//
this.label26.AutoSize = true;
this.label26.Location = new System.Drawing.Point(278, 561);
this.label26.Name = "label26";
this.label26.Size = new System.Drawing.Size(28, 13);
```

```
            this.label26.TabIndex = 71;
            this.label26.Text = "Test";
            //
            // textBox24
            //
            this.textBox24.Location = new System.Drawing.Point(89, 476);
            this.textBox24.Name = "textBox24";
            this.textBox24.Size = new System.Drawing.Size(132, 20);
            this.textBox24.TabIndex = 51;
            this.textBox24.TextChanged += new System.EventHandler(this.textBox24_TextChanged);
            //
            // label27
            //
            this.label27.AutoSize = true;
            this.label27.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label27.Location = new System.Drawing.Point(263, 434);
            this.label27.Name = "label27";
            this.label27.Size = new System.Drawing.Size(237, 13);
            this.label27.TabIndex = 62;
            this.label27.Text = "Defect FixTime Per Phases (time/defect)";
            //
            // label28
            //
            this.label28.AutoSize = true;
            this.label28.Location = new System.Drawing.Point(276, 535);
            this.label28.Name = "label28";
            this.label28.Size = new System.Drawing.Size(44, 13);
            this.label28.TabIndex = 69;
            this.label28.Text = "Compile";
            //
            // textBox25
            //
            this.textBox25.Location = new System.Drawing.Point(89, 502);
            this.textBox25.Name = "textBox25";
            this.textBox25.Size = new System.Drawing.Size(132, 20);
            this.textBox25.TabIndex = 53;
            this.textBox25.TextChanged += new System.EventHandler(this.textBox25_TextChanged);
            //
            // label31
            //
            this.label31.AutoSize = true;
            this.label31.Location = new System.Drawing.Point(276, 482);
            this.label31.Name = "label31";
            this.label31.Size = new System.Drawing.Size(40, 13);
            this.label31.TabIndex = 65;
            this.label31.Text = "Design";
            //
            // textBox28
            //
            this.textBox28.Location = new System.Drawing.Point(89, 580);
            this.textBox28.Name = "textBox28";
            this.textBox28.Size = new System.Drawing.Size(132, 20);
```

```
this.textBox28.TabIndex = 59;
this.textBox28.TextChanged += new System.EventHandler(this.textBox28_TextChanged);
//
// label32
//
this.label32.AutoSize = true;
this.label32.Location = new System.Drawing.Point(276, 508);
this.label32.Name = "label32";
this.label32.Size = new System.Drawing.Size(32, 13);
this.label32.TabIndex = 67;
this.label32.Text = "Code";
//
// textBox29
//
this.textBox29.Location = new System.Drawing.Point(352, 449);
this.textBox29.Name = "textBox29";
this.textBox29.Size = new System.Drawing.Size(132, 20);
this.textBox29.TabIndex = 64;
//
// label33
//
this.label33.AutoSize = true;
this.label33.Location = new System.Drawing.Point(276, 453);
this.label33.Name = "label33";
this.label33.Size = new System.Drawing.Size(48, 13);
this.label33.TabIndex = 63;
this.label33.Text = "Planning";
//
// textBox30
//
this.textBox30.Location = new System.Drawing.Point(352, 475);
this.textBox30.Name = "textBox30";
this.textBox30.Size = new System.Drawing.Size(132, 20);
this.textBox30.TabIndex = 66;
//
// label34
//
this.label34.AutoSize = true;
this.label34.Location = new System.Drawing.Point(18, 585);
this.label34.Name = "label34";
this.label34.Size = new System.Drawing.Size(62, 13);
this.label34.TabIndex = 58;
this.label34.Text = "Postmortem";
//
// textBox31
//
this.textBox31.Location = new System.Drawing.Point(352, 501);
this.textBox31.Name = "textBox31";
this.textBox31.Size = new System.Drawing.Size(132, 20);
this.textBox31.TabIndex = 68;
//
// label35
//
```

```
            this.label35.AutoSize = true;
            this.label35.Location = new System.Drawing.Point(18, 558);
            this.label35.Name = "label35";
            this.label35.Size = new System.Drawing.Size(28, 13);
            this.label35.TabIndex = 56;
            this.label35.Text = "Test";
            //
            // textBox32
            //
            this.textBox32.Location = new System.Drawing.Point(352, 527);
            this.textBox32.Name = "textBox32";
            this.textBox32.Size = new System.Drawing.Size(132, 20);
            this.textBox32.TabIndex = 70;
            //
            // label36
            //
            this.label36.AutoSize = true;
            this.label36.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label36.Location = new System.Drawing.Point(60, 434);
            this.label36.Name = "label36";
            this.label36.Size = new System.Drawing.Size(108, 13);
            this.label36.TabIndex = 47;
            this.label36.Text = "Defects Removed";
            //
            // label37
            //
            this.label37.AutoSize = true;
            this.label37.Location = new System.Drawing.Point(16, 532);
            this.label37.Name = "label37";
            this.label37.Size = new System.Drawing.Size(44, 13);
            this.label37.TabIndex = 54;
            this.label37.Text = "Compile";
            //
            // textBox33
            //
            this.textBox33.Location = new System.Drawing.Point(352, 553);
            this.textBox33.Name = "textBox33";
            this.textBox33.Size = new System.Drawing.Size(132, 20);
            this.textBox33.TabIndex = 72;
            //
            // label40
            //
            this.label40.AutoSize = true;
            this.label40.Location = new System.Drawing.Point(16, 479);
            this.label40.Name = "label40";
            this.label40.Size = new System.Drawing.Size(40, 13);
            this.label40.TabIndex = 50;
            this.label40.Text = "Design";
            //
            // textBox27
            //
            this.textBox27.Location = new System.Drawing.Point(89, 554);
```

```
this.textBox27.Name = "textBox27";
this.textBox27.Size = new System.Drawing.Size(132, 20);
this.textBox27.TabIndex = 57;
this.textBox27.TextChanged += new System.EventHandler(this.textBox27_TextChanged);
//
// label41
//
this.label41.AutoSize = true;
this.label41.Location = new System.Drawing.Point(16, 505);
this.label41.Name = "label41";
this.label41.Size = new System.Drawing.Size(32, 13);
this.label41.TabIndex = 52;
this.label41.Text = "Code";
//
// textBox34
//
this.textBox34.Location = new System.Drawing.Point(352, 579);
this.textBox34.Name = "textBox34";
this.textBox34.Size = new System.Drawing.Size(132, 20);
this.textBox34.TabIndex = 74;
//
// label42
//
this.label42.AutoSize = true;
this.label42.Location = new System.Drawing.Point(16, 450);
this.label42.Name = "label42";
this.label42.Size = new System.Drawing.Size(48, 13);
this.label42.TabIndex = 48;
this.label42.Text = "Planning";
//
// textBox26
//
this.textBox26.Location = new System.Drawing.Point(89, 528);
this.textBox26.Name = "textBox26";
this.textBox26.Size = new System.Drawing.Size(132, 20);
this.textBox26.TabIndex = 55;
this.textBox26.TextChanged += new System.EventHandler(this.textBox26_TextChanged);
//
// label20
//
this.label20.AutoSize = true;
this.label20.Location = new System.Drawing.Point(17, 148);
this.label20.Name = "label20";
this.label20.Size = new System.Drawing.Size(20, 13);
this.label20.TabIndex = 11;
this.label20.Text = "B0";
//
// textBox18
//
this.textBox18.Location = new System.Drawing.Point(343, 231);
this.textBox18.Name = "textBox18";
this.textBox18.Size = new System.Drawing.Size(132, 20);
this.textBox18.TabIndex = 36;
```

```
            this.textBox18.TextChanged += new System.EventHandler(this.textBox18_TextChanged);
            //
            // label21
            //
            this.label21.AutoSize = true;
            this.label21.Location = new System.Drawing.Point(263, 148);
            this.label21.Name = "label21";
            this.label21.Size = new System.Drawing.Size(20, 13);
            this.label21.TabIndex = 13;
            this.label21.Text = "B1";
            //
            // textBox19
            //
            this.textBox19.Location = new System.Drawing.Point(343, 257);
            this.textBox19.Name = "textBox19";
            this.textBox19.Size = new System.Drawing.Size(132, 20);
            this.textBox19.TabIndex = 38;
            this.textBox19.TextChanged += new System.EventHandler(this.textBox19_TextChanged);
            //
            // submitButton
            //
            this.submitButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.submitButton.Location = new System.Drawing.Point(36, 646);
            this.submitButton.Name = "submitButton";
            this.submitButton.Size = new System.Drawing.Size(132, 30);
            this.submitButton.TabIndex = 75;
            this.submitButton.Text = "Submit";
            this.submitButton.UseVisualStyleBackColor = true;
            this.submitButton.Click += new System.EventHandler(this.submitButton_Click);
            //
            // clearAllButton
            //
            this.clearAllButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.clearAllButton.Location = new System.Drawing.Point(188, 646);
            this.clearAllButton.Name = "clearAllButton";
            this.clearAllButton.Size = new System.Drawing.Size(132, 30);
            this.clearAllButton.TabIndex = 76;
            this.clearAllButton.Text = "Clear All";
            this.clearAllButton.UseVisualStyleBackColor = true;
            this.clearAllButton.Click += new System.EventHandler(this.clearAllButton_Click);
            //
            // closeButton
            //
            this.closeButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.closeButton.Location = new System.Drawing.Point(343, 646);
            this.closeButton.Name = "closeButton";
            this.closeButton.Size = new System.Drawing.Size(132, 30);
            this.closeButton.TabIndex = 77;
            this.closeButton.Text = "Close";
            this.closeButton.UseVisualStyleBackColor = true;
```

```
            this.closeButton.Click += new System.EventHandler(this.closeButton_Click);
            //
            // label29
            //
            this.label29.AutoSize = true;
            this.label29.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label29.Location = new System.Drawing.Point(201, 73);
            this.label29.Name = "label29";
            this.label29.Size = new System.Drawing.Size(123, 13);
            this.label29.TabIndex = 4;
            this.label29.Text = "PSP Historical Data:";
            //
            // textBox4
            //
            this.textBox4.Location = new System.Drawing.Point(110, 115);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(132, 20);
            this.textBox4.TabIndex = 8;
            //
            // label30
            //
            this.label30.AutoSize = true;
            this.label30.Location = new System.Drawing.Point(16, 119);
            this.label30.Name = "label30";
            this.label30.Size = new System.Drawing.Size(88, 13);
            this.label30.TabIndex = 7;
            this.label30.Text = "Lower LOC/Hour";
            //
            // textBox35
            //
            this.textBox35.Location = new System.Drawing.Point(88, 606);
            this.textBox35.Name = "textBox35";
            this.textBox35.Size = new System.Drawing.Size(132, 20);
            this.textBox35.TabIndex = 61;
            //
            // AgentForm
            //
            this.ClientSize = new System.Drawing.Size(507, 688);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.label30);
            this.Controls.Add(this.label29);
            this.Controls.Add(this.closeButton);
            this.Controls.Add(this.clearAllButton);
            this.Controls.Add(this.submitButton);
            this.Controls.Add(this.label21);
            this.Controls.Add(this.textBox19);
            this.Controls.Add(this.label20);
            this.Controls.Add(this.textBox18);
            this.Controls.Add(this.label25);
            this.Controls.Add(this.textBox23);
            this.Controls.Add(this.label26);
            this.Controls.Add(this.textBox24);
```

```
this.Controls.Add(this.label27);
this.Controls.Add(this.label28);
this.Controls.Add(this.textBox25);
this.Controls.Add(this.label31);
this.Controls.Add(this.textBox28);
this.Controls.Add(this.label32);
this.Controls.Add(this.textBox29);
this.Controls.Add(this.label33);
this.Controls.Add(this.textBox30);
this.Controls.Add(this.label34);
this.Controls.Add(this.textBox31);
this.Controls.Add(this.label35);
this.Controls.Add(this.textBox32);
this.Controls.Add(this.label36);
this.Controls.Add(this.label37);
this.Controls.Add(this.textBox33);
this.Controls.Add(this.label40);
this.Controls.Add(this.textBox27);
this.Controls.Add(this.label41);
this.Controls.Add(this.textBox34);
this.Controls.Add(this.label42);
this.Controls.Add(this.textBox26);
this.Controls.Add(this.label16);
this.Controls.Add(this.textBox15);
this.Controls.Add(this.label17);
this.Controls.Add(this.textBox16);
this.Controls.Add(this.label18);
this.Controls.Add(this.label19);
this.Controls.Add(this.textBox17);
this.Controls.Add(this.label22);
this.Controls.Add(this.textBox20);
this.Controls.Add(this.label23);
this.Controls.Add(this.textBox21);
this.Controls.Add(this.label24);
this.Controls.Add(this.textBox22);
this.Controls.Add(this.label14);
this.Controls.Add(this.textBox13);
this.Controls.Add(this.label15);
this.Controls.Add(this.textBox14);
this.Controls.Add(this.label13);
this.Controls.Add(this.label9);
this.Controls.Add(this.textBox9);
this.Controls.Add(this.label10);
this.Controls.Add(this.textBox10);
this.Controls.Add(this.label11);
this.Controls.Add(this.textBox11);
this.Controls.Add(this.label12);
this.Controls.Add(this.textBox12);
this.Controls.Add(this.label5);
this.Controls.Add(this.textBox5);
this.Controls.Add(this.label6);
this.Controls.Add(this.textBox6);
this.Controls.Add(this.label7);
```

```
    this.Controls.Add(this.textBox7);
    this.Controls.Add(this.label8);
    this.Controls.Add(this.textBox8);
    this.Controls.Add(this.label4);
    this.Controls.Add(this.textBox35);
    this.Controls.Add(this.label3);
    this.Controls.Add(this.textBox3);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.textBox2);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.textBox1);
    this.Name = "AgentForm";
    this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
    this.Text = "Employee Form";
    this.ResumeLayout(false);
    this.PerformLayout();

}

#endregion

private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.TextBox textBox2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.TextBox textBox3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.TextBox textBox5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.TextBox textBox6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.TextBox textBox7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.TextBox textBox8;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.TextBox textBox9;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.TextBox textBox10;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.TextBox textBox11;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.TextBox textBox12;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.TextBox textBox13;
private System.Windows.Forms.Label label15;
private System.Windows.Forms.TextBox textBox14;
private System.Windows.Forms.Label label16;
private System.Windows.Forms.TextBox textBox15;
private System.Windows.Forms.Label label17;
private System.Windows.Forms.TextBox textBox16;
```

```
        private System.Windows.Forms.Label label18;
        private System.Windows.Forms.Label label19;
        private System.Windows.Forms.TextBox textBox17;
        private System.Windows.Forms.Label label22;
        private System.Windows.Forms.TextBox textBox20;
        private System.Windows.Forms.Label label23;
        private System.Windows.Forms.TextBox textBox21;
        private System.Windows.Forms.Label label24;
        private System.Windows.Forms.TextBox textBox22;
        private System.Windows.Forms.Label label25;
        private System.Windows.Forms.TextBox textBox23;
        private System.Windows.Forms.Label label26;
        private System.Windows.Forms.TextBox textBox24;
        private System.Windows.Forms.Label label27;
        private System.Windows.Forms.Label label28;
        private System.Windows.Forms.TextBox textBox25;
        private System.Windows.Forms.Label label31;
        private System.Windows.Forms.TextBox textBox28;
        private System.Windows.Forms.Label label32;
        private System.Windows.Forms.TextBox textBox29;
        private System.Windows.Forms.Label label33;
        private System.Windows.Forms.TextBox textBox30;
        private System.Windows.Forms.Label label34;
        private System.Windows.Forms.TextBox textBox31;
        private System.Windows.Forms.Label label35;
        private System.Windows.Forms.TextBox textBox32;
        private System.Windows.Forms.Label label36;
        private System.Windows.Forms.Label label37;
        private System.Windows.Forms.TextBox textBox33;
        private System.Windows.Forms.Label label40;
        private System.Windows.Forms.TextBox textBox27;
        private System.Windows.Forms.Label label41;
        private System.Windows.Forms.TextBox textBox34;
        private System.Windows.Forms.Label label42;
        private System.Windows.Forms.TextBox textBox26;
        private System.Windows.Forms.Label label20;
        private System.Windows.Forms.TextBox textBox18;
        private System.Windows.Forms.Label label21;
        private System.Windows.Forms.TextBox textBox19;
        private System.Windows.Forms.Button submitButton;
        private System.Windows.Forms.Button clearAllButton;
        private System.Windows.Forms.Button closeButton;
        private System.Windows.Forms.Label label29;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.Label label30;
        private System.Windows.Forms.TextBox textBox35;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;


namespace SDPTool
{
   public partial class AgentForm : Form
   {
      string folderPath = "C:\\SDPTool\\Results\\";

      public string FolderPath
      {
         get  //get accessor method
         {
            return folderPath;
         }
         set   //set accessor method
         {
            folderPath = value;
         }
      }

      public AgentForm()
      {
         InitializeComponent();
      }

      private void submitButton_Click(object sender, EventArgs e)
      {
         try
         {
            FileStream file;
            StreamWriter sw;
            file = new FileStream(folderPath + "Employees.txt", FileMode.Append, FileAccess.Write);
            sw = new StreamWriter(file);
            sw.Write(textBox1.Text); sw.Write(", ");
            sw.Write(textBox2.Text); sw.Write(", ");
            sw.Write(textBox3.Text); sw.Write(", ");
            sw.Write(textBox35.Text); sw.Write(", ");
            sw.Write(textBox4.Text); sw.Write(", ");
            sw.Write(textBox5.Text); sw.Write(", ");
            sw.Write(textBox6.Text); sw.Write(", ");
            sw.Write(textBox7.Text); sw.Write(", ");
            sw.Write(textBox8.Text); sw.Write(", ");
            sw.Write(textBox9.Text); sw.Write(", ");
            sw.Write(textBox10.Text); sw.Write(", ");
```

```
sw.Write(textBox11.Text); sw.Write(", ");
sw.Write(textBox12.Text); sw.Write(", ");
sw.Write(textBox13.Text); sw.Write(", ");
sw.Write(textBox14.Text); sw.Write(", ");
sw.Write(textBox15.Text); sw.Write(", ");
sw.Write(textBox16.Text); sw.Write(", ");
sw.Write(textBox17.Text); sw.Write(", ");
sw.Write(textBox18.Text); sw.Write(", ");
sw.Write(textBox19.Text); sw.Write(", ");
sw.Write(textBox20.Text); sw.Write(", ");
sw.Write(textBox21.Text); sw.Write(", ");
sw.Write(textBox22.Text); sw.Write(", ");
sw.Write(textBox23.Text); sw.Write(", ");
sw.Write(textBox24.Text); sw.Write(", ");
sw.Write(textBox25.Text); sw.Write(", ");
sw.Write(textBox26.Text); sw.Write(", ");
sw.Write(textBox27.Text); sw.Write(", ");
sw.Write(textBox28.Text); sw.Write(", ");
sw.Write(textBox29.Text); sw.Write(", ");
sw.Write(textBox30.Text); sw.Write(", ");
sw.Write(textBox31.Text); sw.Write(", ");
sw.Write(textBox32.Text); sw.Write(", ");
sw.Write(textBox33.Text); sw.Write(", ");
sw.Write(textBox34.Text);
sw.WriteLine();
sw.Close();
file.Close();
textBox1.Clear();
textBox2.Clear();
textBox3.Clear();
textBox35.Clear();
textBox5.Clear();
textBox6.Clear();
textBox7.Clear();
textBox8.Clear();
textBox9.Clear();
textBox10.Clear();
textBox11.Clear();
textBox12.Clear();
textBox13.Clear();
textBox14.Clear();
textBox15.Clear();
textBox16.Clear();
textBox17.Clear();
textBox18.Clear();
textBox19.Clear();
textBox20.Clear();
textBox21.Clear();
textBox22.Clear();
textBox23.Clear();
textBox24.Clear();
textBox25.Clear();
textBox26.Clear();
```

```
            textBox27.Clear();
            textBox28.Clear();
            textBox29.Clear();
            textBox30.Clear();
            textBox31.Clear();
            textBox32.Clear();
            textBox33.Clear();
            textBox34.Clear();
            textBox4.Clear();
        }
        catch
        {
            // Display a message box and clear the contents if not a number.
            MessageBox.Show("The PSP values are not valid. Please check the values again");
        }
    }

    private void clearAllButton_Click(object sender, EventArgs e)
    {
        textBox1.Clear();
        textBox2.Clear();
        textBox3.Clear();
        textBox35.Clear();
        textBox5.Clear();
        textBox6.Clear();
        textBox7.Clear();
        textBox8.Clear();
        textBox9.Clear();
        textBox10.Clear();
        textBox11.Clear();
        textBox12.Clear();
        textBox13.Clear();
        textBox14.Clear();
        textBox15.Clear();
        textBox16.Clear();
        textBox17.Clear();
        textBox18.Clear();
        textBox19.Clear();
        textBox20.Clear();
        textBox21.Clear();
        textBox22.Clear();
        textBox23.Clear();
        textBox24.Clear();
        textBox25.Clear();
        textBox26.Clear();
        textBox27.Clear();
        textBox28.Clear();
        textBox29.Clear();
        textBox30.Clear();
        textBox31.Clear();
        textBox32.Clear();
        textBox33.Clear();
        textBox34.Clear();
```

```
            textBox4.Clear();
        }

        private void closeButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void textBox17_TextChanged(object sender, EventArgs e)
        {
            if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
                {
                    textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
                }
            }
            else
                textBox6.Text = null;
        }

        private void textBox18_TextChanged(object sender, EventArgs e)
        {
            if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
                {
                    textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
                }
            }
            else
                textBox6.Text = null;
        }

        private void textBox19_TextChanged(object sender, EventArgs e)
        {
            if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
            {
```

```
            if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
            {
               textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
            }
         }
         else
            textBox6.Text = null;
      }

      private void textBox20_TextChanged(object sender, EventArgs e)
      {
         if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
         {
            if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
            {
               textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
            }
         }
         else
            textBox6.Text = null;
      }

      private void textBox21_TextChanged(object sender, EventArgs e)
      {
         if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
         {
            if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
            {
               textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
            }
         }
         else
            textBox6.Text = null;
      }

      private void textBox22_TextChanged(object sender, EventArgs e)
```

```
        {
            if ((textBox17.TextLength > 0) && (textBox18.TextLength > 0) && (textBox19.TextLength > 0)
&& (textBox20.TextLength > 0) && (textBox21.TextLength > 0) && (textBox22.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox17.Text) >= 0) && (Convert.ToDouble(textBox18.Text) >= 0)
&& (Convert.ToDouble(textBox19.Text) >= 0) && (Convert.ToDouble(textBox20.Text) >= 0) &&
(Convert.ToDouble(textBox21.Text) >= 0) && (Convert.ToDouble(textBox22.Text) >= 0))
                {
                    textBox6.Text = Convert.ToString(Convert.ToDouble(textBox17.Text) +
Convert.ToDouble(textBox18.Text) + Convert.ToDouble(textBox19.Text) +
Convert.ToDouble(textBox20.Text) + Convert.ToDouble(textBox21.Text) +
Convert.ToDouble(textBox22.Text));
                }
            }
            else
                textBox6.Text = null;
        }

        private void textBox23_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
                textBox35.Text = null;
        }

        private void textBox24_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
```

```
            textBox35.Text = null;
        }

        private void textBox25_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
                textBox35.Text = null;
        }

        private void textBox26_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
                textBox35.Text = null;
        }

        private void textBox27_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
```

```
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
                textBox35.Text = null;
        }

        private void textBox28_TextChanged(object sender, EventArgs e)
        {
            if ((textBox23.TextLength > 0) && (textBox24.TextLength > 0) && (textBox25.TextLength > 0)
&& (textBox26.TextLength > 0) && (textBox27.TextLength > 0) && (textBox28.TextLength > 0))
            {
                if ((Convert.ToDouble(textBox23.Text) >= 0) && (Convert.ToDouble(textBox24.Text) >= 0)
&& (Convert.ToDouble(textBox25.Text) >= 0) && (Convert.ToDouble(textBox26.Text) >= 0) &&
(Convert.ToDouble(textBox27.Text) >= 0) && (Convert.ToDouble(textBox28.Text) >= 0))
                {
                    textBox35.Text = Convert.ToString(Convert.ToDouble(textBox23.Text) +
Convert.ToDouble(textBox24.Text) + Convert.ToDouble(textBox25.Text) +
Convert.ToDouble(textBox26.Text) + Convert.ToDouble(textBox27.Text) +
Convert.ToDouble(textBox28.Text));
                }
            }
            else
                textBox35.Text = null;
        }
    }
}
```

```
using System;
using System.IO;
using System.Threading;

namespace SDPTool
{
   class Agent
   {
      static int TOTALFEATURES = 35;
      static int MONTECARLO = 10;

      int id;
      string name;
      double pspLocDeveloped;
      double pspDefectsRemoved;
      double pspDefectsInjected;
      double pspLowerLocPerHour;
      double pspUpperLocPerHour;
      double pspDefectsPerLoc;
      double pspB0;
      double pspB1;
      double pspPlanningTime;
      double pspDesignTime;
      double pspDesignReviewTime;
      double pspCodeTime;
      double pspCodeReviewTime;
      double pspCompileTime;
      double pspTestTime;
      double pspRefactorTime;
      double pspPlanningDefects;
      double pspDesignDefects;
      double pspCodeDefects;
      double pspCompileDefects;
      double pspTestDefects;
      double pspRefactorDefects;
      double pspDefectsRemovedPlanning;
      double pspDefectsRemovedDesign;
      double pspDefectsRemovedCode;
      double pspDefectsRemovedCompile;
      double pspDefectsRemovedTest;
      double pspDefectsRemovedRefactor;
      double pspPlanningFixTime;
      double pspDesignFixTime;
      double pspCodeFixTime;
      double pspCompileFixTime;
      double pspTestFixTime;
      double pspRefactorFixTime;
      string folderPath = "C:\\SDPTool\\Results\\";
      string fileName = null;
      File outFile = null;
      string activityType = null;
      Team parent;
```

```
double totalLocPlanned = 0;
double totalLocDesigned = 0;
double totalLocDesignReviewed = 0;
double totalLocWritten = 0;
double totalLocCodeReviewed = 0;
double totalLocCompiled = 0;
double totalLocTested = 0;
double totalLocRefactored = 0;
double totalLocCompleted = 0;
double totalDefectsRemoved = 0;
double totalDefectsIntroduced = 0;
double newLoc = 0;
double locPlanned = 0;
double locDesigned = 0;
double locDesignReviewed = 0;
double locWritten = 0;
double locCodeReviewed = 0;
double locCompiled = 0;
double locTested = 0;
double locRefactored = 0;
double locCompleted = 0;
double defectsIntroduced = 0;
double defectsRemoved = 0;
double currentDefects = 0;
Random rand = new Random();

public static void EnsureDirectory(System.IO.DirectoryInfo oDirInfo)
{
   if (oDirInfo.Parent != null)
      EnsureDirectory(oDirInfo.Parent);
   if (!oDirInfo.Exists)
   {
      oDirInfo.Create();
   }
}

public int Id
{
   get  //get accessor method
   {
      return id;
   }
   set   //set accessor method
   {
      id = value;
   }
}

public string Name
{
   get  //get accessor method
   {
      return name;
```

```
    }
    set   //set accessor method
    {
        name = value;
    }
}

public Team Parent
{
    get  //get accessor method
    {
        return parent;
    }
    set   //set accessor method
    {
        parent = value;
    }
}

public double PspB0
{
    get  //get accessor method
    {
        return pspB0;
    }
}

public double PspB1
{
    get  //get accessor method
    {
        return pspB1;
    }
}

public double PspLowerLocPerHour
{
    get  //get accessor method
    {
        return pspLowerLocPerHour;
    }
}

public double PspUpperLocPerHour
{
    get  //get accessor method
    {
        return pspUpperLocPerHour;
    }
}

public double LocPlanned
{
```

```
    get  //get accessor method
    {
        return locPlanned;
    }
    set   //set accessor method
    {
        locPlanned = value;
    }
}

public double LocDesigned
{
    get  //get accessor method
    {
        return locDesigned;
    }
    set   //set accessor method
    {
        locDesigned = value;
    }
}

public double LocDesignReviewed
{
    get  //get accessor method
    {
        return locDesignReviewed;
    }
    set   //set accessor method
    {
        locDesignReviewed = value;
    }
}

public double LocWritten
{
    get  //get accessor method
    {
        return locWritten;
    }
    set   //set accessor method
    {
        locWritten = value;
    }
}

public double LocCodeReviewed
{
    get  //get accessor method
    {
        return locCodeReviewed;
    }
    set   //set accessor method
```

```
    {
        locCodeReviewed = value;
    }
}

public double LocCompiled
{
    get  //get accessor method
    {
        return locCompiled;
    }
    set   //set accessor method
    {
        locCompiled = value;
    }
}

public double LocTested
{
    get  //get accessor method
    {
        return locTested;
    }
    set   //set accessor method
    {
        locTested = value;
    }
}

public double LocRefactored
{
    get  //get accessor method
    {
        return locRefactored;
    }
    set   //set accessor method
    {
        locRefactored = value;
    }
}

public double LocCompleted
{
    get  //get accessor method
    {
        return locCompleted;
    }
    set   //set accessor method
    {
        locCompleted = value;
    }
}
```

```
public double CurrentDefects
{
   get  //get accessor method
   {
      return currentDefects;
   }
   set   //set accessor method
   {
      currentDefects = value;
   }
}

public double DefectsIntroduced
{
   get  //get accessor method
   {
      return defectsIntroduced;
   }
   set   //set accessor method
   {
      defectsIntroduced = value;
   }
}

public double DefectsRemoved
{
   get  //get accessor method
   {
      return defectsRemoved;
   }
   set   //set accessor method
   {
      defectsRemoved = value;
   }
}

public double TotalLocPlanned
{
   get  //get accessor method
   {
      return totalLocPlanned;
   }
   set   //set accessor method
   {
      totalLocPlanned = value;
   }
}

public double TotalLocDesigned
{
   get  //get accessor method
   {
      return totalLocDesigned;
```

```
    }
    set   //set accessor method
    {
        totalLocDesigned = value;
    }
}

public double TotalLocDesignReviewed
{
    get  //get accessor method
    {
        return totalLocDesignReviewed;
    }
    set   //set accessor method
    {
        totalLocDesignReviewed = value;
    }
}

public double TotalLocWritten
{
    get  //get accessor method
    {
        return totalLocWritten;
    }
    set   //set accessor method
    {
        totalLocWritten = value;
    }
}

public double TotalLocCodeReviewed
{
    get  //get accessor method
    {
        return totalLocCodeReviewed;
    }
    set   //set accessor method
    {
        totalLocCodeReviewed = value;
    }
}

public double TotalLocCompiled
{
    get  //get accessor method
    {
        return totalLocCompiled;
    }
    set   //set accessor method
    {
        totalLocCompiled = value;
    }
```

```
}

public double TotalLocTested
{
   get  //get accessor method
   {
      return totalLocTested;
   }
   set   //set accessor method
   {
      totalLocTested = value;
   }
}

public double TotalLocRefactored
{
   get  //get accessor method
   {
      return totalLocRefactored;
   }
   set   //set accessor method
   {
      totalLocRefactored = value;
   }
}

public double TotalLocCompleted
{
   get  //get accessor method
   {
      return totalLocCompleted;
   }
   set   //set accessor method
   {
      totalLocCompleted = value;
   }
}

public double TotalDefectsIntroduced
{
   get  //get accessor method
   {
      return totalDefectsIntroduced;
   }
   set   //set accessor method
   {
      totalDefectsIntroduced = value;
   }
}

public double TotalDefectsRemoved
{
   get  //get accessor method
```

```
    {
        return totalDefectsRemoved;
    }
    set   //set accessor method
    {
        totalDefectsRemoved = value;
    }
}

public string ActivityType
{
    get  //get accessor method
    {
        return activityType;
    }
    set   //set accessor method
    {
        activityType = value;
    }
}

public void Run()
{
    //Initialize for this Run...
    locPlanned = 0;
    locDesigned = 0;
    locDesignReviewed = 0;
    locWritten = 0;
    locCodeReviewed = 0;
    locCompiled = 0;
    locTested = 0;
    locRefactored = 0;
    locCompleted = 0;
    //Start the activity...
    switch (activityType)
    {
        case "Planning":
            Planning();
            break;
        case "Design":
            Design();
            break;
        case "DesignReview":
            DesignReview();
            break;
        case "Code":
            Code();
            break;
        case "CodeReview":
            CodeReview();
            break;
        case "Compile":
            Compile();
```

```
                break;
            case "Test":
                Test();
                break;
            case "Debug":
                Debug();
                break;
            case "Refactor":
                Refactor();
                break;
            case "Idle":
                Idle();
                break;
        }
        if (activityType.Equals("Refactor") && (currentDefects <= 0))
            locCompleted = locRefactored;

        totalLocPlanned += locPlanned;
        totalLocDesigned += locDesigned;
        totalLocDesignReviewed += locDesignReviewed;
        totalLocWritten += locWritten;
        totalLocCodeReviewed += locCodeReviewed;
        totalLocCompiled += locCompiled;
        totalLocTested += locTested;
        totalLocRefactored += locRefactored;
        totalLocCompleted += locCompleted;
        if (outFile != null)
        {
            outFile.WriteLine(string.Concat("ID of ", name, "'s thread is ",
Convert.ToString(Thread.CurrentThread.GetHashCode())));
        }
    }



    private void Idle()
    {
        //Do nothing!!!
    }

    private void Planning()
    {
        newLoc = 0;
        defectsIntroduced = 0;
        defectsRemoved = 0;

        if (parent.Story.LocPlanned < parent.Story.Size)
        {
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                newLoc += (pspCodeTime / pspPlanningTime) * (pspLowerLocPerHour + rand.NextDouble()
* (pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
```

```
                defectsIntroduced += ((pspPlanningDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                if (pspPlanningFixTime > 0)
                    defectsRemoved += (1 / pspPlanningFixTime) * (rand.NextDouble()); // # of defects
removed per min.
            }
            newLoc = newLoc / MONTECARLO;
            defectsIntroduced = defectsIntroduced / MONTECARLO;
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            locPlanned += newLoc;
            totalDefectsIntroduced += defectsIntroduced;
            totalDefectsRemoved += defectsRemoved;
            currentDefects += defectsIntroduced - defectsRemoved;
        }
        else if (parent.Story.CurrentDefects > 0)
        {
            // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
            //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                if (pspPlanningFixTime > 0)
                    defectsRemoved += (1 / pspPlanningFixTime) * (rand.NextDouble());
            }
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
        }
    }

    private void Design()
    {
        newLoc = 0;
        defectsIntroduced = 0;
        defectsRemoved = 0;

        if (parent.Story.LocDesigned < parent.Story.Size)
        {
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                newLoc += (pspCodeTime / pspDesignTime) * (pspLowerLocPerHour + rand.NextDouble() *
(pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
                defectsIntroduced += ((pspDesignDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                if (pspDesignFixTime > 0)
                    defectsRemoved += (1 / pspDesignFixTime) * (rand.NextDouble());
            }
            newLoc = newLoc / MONTECARLO;
```

```
            defectsIntroduced = defectsIntroduced / MONTECARLO;
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            locDesigned += newLoc;
            totalDefectsIntroduced += defectsIntroduced;
            totalDefectsRemoved += defectsRemoved;
            currentDefects += defectsIntroduced - defectsRemoved;
         }
         else if (parent.Story.CurrentDefects > 0)
         {
            // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
            //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
               if (pspDesignFixTime > 0)
                  defectsRemoved += (1 / pspDesignFixTime) * (rand.NextDouble());

            }
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
         }
      }

      private void DesignReview()
      {
         newLoc = 0;
         defectsIntroduced = 0;
         defectsRemoved = 0;

         if (parent.Story.LocDesignReviewed < parent.Story.Size)
         {
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
               newLoc += (pspCodeTime / pspDesignReviewTime) * (pspLowerLocPerHour +
rand.NextDouble() * (pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
               if (pspDesignFixTime > 0)
                  defectsRemoved += (1 / pspDesignFixTime) * (rand.NextDouble());
            }
            newLoc = newLoc / MONTECARLO;
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            locDesignReviewed += newLoc;
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
         }
      }
```

```
        private void Code()
        {
            newLoc = 0;
            defectsIntroduced = 0;
            defectsRemoved = 0;

            if (parent.Story.LocWritten < parent.Story.Size)
            {
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
                    newLoc += (pspLowerLocPerHour + rand.NextDouble() * (pspUpperLocPerHour -
pspLowerLocPerHour)) / 60;
                    defectsIntroduced += ((pspCodeDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                    if (pspCodeFixTime > 0)
                        defectsRemoved += (1 / pspCodeFixTime) * (rand.NextDouble());
                }
                newLoc = newLoc / MONTECARLO;
                defectsIntroduced = defectsIntroduced / MONTECARLO;
                defectsRemoved = defectsRemoved / MONTECARLO;
                //Monte-Carlo ends.

                locWritten += newLoc;
                totalDefectsIntroduced += defectsIntroduced;
                totalDefectsRemoved += defectsRemoved;
                currentDefects += defectsIntroduced - defectsRemoved;
            }
            else if (parent.Story.CurrentDefects > 0)
            {
                // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
                //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
                    if (pspCodeFixTime > 0)
                        defectsRemoved += (1 / pspCodeFixTime) * (rand.NextDouble());
                }
                defectsRemoved = defectsRemoved / MONTECARLO;
                //Monte-Carlo ends.
                totalDefectsRemoved += defectsRemoved;
                currentDefects -= defectsRemoved;
            }
        }

        private void Debug()
        {
            defectsIntroduced = 0;
            defectsRemoved = 0;

            //Monte-Carlo Starts...
```

```
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                if (pspCodeFixTime > 0)
                    defectsRemoved += (1 / pspCodeFixTime) * (rand.NextDouble()); // # of defects removed in 1
min.

            }
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo Ends.
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
            if (parent.Story.CurrentDefects <= 0)
            {
                currentDefects = 0;
            }
            else
            {
                totalDefectsRemoved += defectsRemoved;
            }
        }

        private void CodeReview()
        {
            newLoc = 0;
            defectsIntroduced = 0;
            defectsRemoved = 0;

            if (parent.Story.LocCodeReviewed < parent.Story.Size)
            {
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
                    newLoc += (pspCodeTime / pspCodeReviewTime) * (pspLowerLocPerHour +
rand.NextDouble() * (pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
                    if (pspCodeFixTime > 0)
                        defectsRemoved += (1 / pspCodeFixTime) * (rand.NextDouble());
                }
                newLoc = newLoc / MONTECARLO;
                defectsRemoved = defectsRemoved / MONTECARLO;
                //Monte-Carlo ends.

                locCodeReviewed += newLoc;
                totalDefectsRemoved += defectsRemoved;
                currentDefects -= defectsRemoved;
            }
            else if (parent.Story.CurrentDefects > 0)
            {
                // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
                //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
```

```
                if (pspCodeFixTime > 0)
                    defectsRemoved += (1 / pspCodeFixTime) * (rand.NextDouble()); // # of defects removed
in 1 min.

            }
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
        }
    }

    private void Compile()
    {
        newLoc = 0;
        defectsIntroduced = 0;
        defectsRemoved = 0;

        if (parent.Story.LocCompiled < parent.Story.Size)
        {
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                newLoc += (pspCodeTime / pspCompileTime) * (pspLowerLocPerHour + rand.NextDouble()
* (pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
                defectsIntroduced += ((pspCompileDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                if (pspCompileFixTime > 0)
                    defectsRemoved += (1 / pspCompileFixTime) * (rand.NextDouble());
            }
            newLoc = newLoc / MONTECARLO;
            defectsIntroduced = defectsIntroduced / MONTECARLO;
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            locCompiled += newLoc;
            totalDefectsIntroduced += defectsIntroduced;
            totalDefectsRemoved += defectsRemoved;
            currentDefects += defectsIntroduced - defectsRemoved;
        }
        else if (parent.Story.CurrentDefects > 0)
        {
            // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
            //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                if (pspCompileFixTime > 0)
                    defectsRemoved += (1 / pspCompileFixTime) * (rand.NextDouble());

            }
```

```
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
        }

    }

    private void Test()
    {
        newLoc = 0;
        defectsIntroduced = 0;
        defectsRemoved = 0;

        if (parent.Story.LocTested < parent.Story.Size)
        {
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                newLoc += (pspCodeTime / pspTestTime) * (pspLowerLocPerHour + rand.NextDouble() *
(pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
                defectsIntroduced += ((pspTestDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                if (pspTestFixTime > 0)
                    defectsRemoved += (1 / pspTestFixTime) * (rand.NextDouble());
            }
            newLoc = newLoc / MONTECARLO;
            defectsIntroduced = defectsIntroduced / MONTECARLO;
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.

            locTested += newLoc;
            totalDefectsIntroduced += defectsIntroduced;
            totalDefectsRemoved += defectsRemoved;
            currentDefects += defectsIntroduced - defectsRemoved;
        }
        else if (parent.Story.CurrentDefects > 0)
        {
            // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
            //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
            //Monte-Carlo Starts...
            for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
            {
                if (pspTestFixTime > 0)
                    defectsRemoved += (1 / pspTestFixTime) * (rand.NextDouble());

            }
            defectsRemoved = defectsRemoved / MONTECARLO;
            //Monte-Carlo ends.
            totalDefectsRemoved += defectsRemoved;
            currentDefects -= defectsRemoved;
        }
```

```
        }

        private void Refactor()
        {
            newLoc = 0;
            defectsIntroduced = 0;
            defectsRemoved = 0;

            if (parent.Story.LocRefactored < parent.Story.Size)
            {
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
                    newLoc += (pspCodeTime / pspRefactorTime) * (pspLowerLocPerHour + rand.NextDouble()
* (pspUpperLocPerHour - pspLowerLocPerHour)) / 60;
                    defectsIntroduced += ((pspRefactorDefects / 100) * pspDefectsPerLoc * newLoc) *
(rand.NextDouble());
                    if (pspRefactorFixTime > 0)
                        defectsRemoved += (1 / pspRefactorFixTime) * (rand.NextDouble());
                }
                newLoc = newLoc / MONTECARLO;
                defectsIntroduced = defectsIntroduced / MONTECARLO;
                defectsRemoved = defectsRemoved / MONTECARLO;
                //Monte-Carlo ends.

                locRefactored += newLoc;
                totalDefectsIntroduced += defectsIntroduced;
                totalDefectsRemoved += defectsRemoved;
                currentDefects += defectsIntroduced - defectsRemoved;
            }
            else if (parent.Story.CurrentDefects > 0)
            {
                // (Ratio) defects removed : total defects introduced = defectsRemoved : defectsPerLoc *
locDeveloped;
                //defects removed in code = (defectsRemovedCode/100)*defectsToBeRemovedInCode;
                //Monte-Carlo Starts...
                for (int monteCarlo = 0; monteCarlo < MONTECARLO; monteCarlo++)
                {
                    if (pspRefactorFixTime > 0)
                        defectsRemoved += (1 / pspRefactorFixTime) * (rand.NextDouble());

                }
                defectsRemoved = defectsRemoved / MONTECARLO;
                //Monte-Carlo ends.
                totalDefectsRemoved += defectsRemoved;
                currentDefects -= defectsRemoved;
            }
        }

        public Agent(string line, char[] delimeter)
        {
            string[] features = new string[TOTALFEATURES + 1];
            string[] words = new string[TOTALFEATURES];
```

```
    words = line.Split(delimeter);
    int i = 0;
    foreach (string word in words)
    {
      features[++i] = word.Trim();
    }
    if (i == TOTALFEATURES)
    {
      id = Int32.Parse(features[1]);
      name = features[2];
      pspLocDeveloped = Double.Parse(features[3]);
      pspDefectsRemoved = Double.Parse(features[4]);
      pspLowerLocPerHour = Double.Parse(features[5]);
      pspUpperLocPerHour = Double.Parse(features[6]);
      pspDefectsInjected = Double.Parse(features[7]);
      pspDefectsPerLoc = pspDefectsInjected / pspLocDeveloped;
      pspB0 = Double.Parse(features[8]);
      pspB1 = Double.Parse(features[9]);
      pspPlanningTime = Double.Parse(features[10]);
      pspDesignTime = Double.Parse(features[11]);
      pspDesignReviewTime = Double.Parse(features[12]);
      pspCodeTime = Double.Parse(features[13]);
      pspCodeReviewTime = Double.Parse(features[14]);
      pspCompileTime = Double.Parse(features[15]);
      pspTestTime = Double.Parse(features[16]);
      pspRefactorTime = Double.Parse(features[17]);
      pspPlanningDefects = Double.Parse(features[18]) * 100 / pspDefectsInjected;
      pspDesignDefects = Double.Parse(features[19]) * 100 / pspDefectsInjected;
      pspCodeDefects = Double.Parse(features[20]) * 100 / pspDefectsInjected;
      pspCompileDefects = Double.Parse(features[21]) * 100 / pspDefectsInjected;
      pspTestDefects = Double.Parse(features[22]) * 100 / pspDefectsInjected;
      pspRefactorDefects = Double.Parse(features[23]) * 100 / pspDefectsInjected;
      pspDefectsRemovedPlanning = Double.Parse(features[24]) * 100 / pspDefectsRemoved;
      pspDefectsRemovedDesign = Double.Parse(features[25]) * 100 / pspDefectsRemoved;
      pspDefectsRemovedCode = Double.Parse(features[26]) * 100 / pspDefectsRemoved;
      pspDefectsRemovedCompile = Double.Parse(features[27]) * 100 / pspDefectsRemoved;
      pspDefectsRemovedTest = Double.Parse(features[28]) * 100 / pspDefectsRemoved;
      pspDefectsRemovedRefactor = Double.Parse(features[29]) * 100 / pspDefectsRemoved;
      pspPlanningFixTime = Double.Parse(features[30]);
      pspDesignFixTime = Double.Parse(features[31]);
      pspCodeFixTime = Double.Parse(features[32]);
      pspCompileFixTime = Double.Parse(features[33]);
      pspTestFixTime = Double.Parse(features[34]);
      pspRefactorFixTime = Double.Parse(features[35]);

      //To write things individually for each agent
      EnsureDirectory(new System.IO.DirectoryInfo(folderPath + "\\Employees"));
      folderPath += "\\Employees\\";
      fileName = string.Concat(folderPath, name, ".txt");
      outFile = new File(fileName);
      activityType = null;
    }
  }
```

```
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace SDPTool
{

   public class AboutForm : Form
   {
      private System.Windows.Forms.Button buttonAboutOK;
      private System.Windows.Forms.GroupBox groupAbout;
      private System.Windows.Forms.Label labelAbout;
      private System.Windows.Forms.LinkLabel linkLabelAbout;
      private PictureBox pictureBoxAbout;

      private System.ComponentModel.Container components = null;

      public AboutForm()
      {
         InitializeComponent();
         this.Text = " About: Software Developement Process Simulation";
         this.linkLabelAbout.Text = "http://www.eng.auburn.edu/users/agarwra/";
         this.labelAbout.Text = "Software Developement Process Simulation" + "\n" +
                        "SDPTool" + " Version 1" + "\n" +
                        "Copyright ©  2007 by author: Ravikant Agarwal" + "\n\n\n" +
                        "Ravikant Agarwal, Doctoral Student" + "\n" +
                        "Computer Science and Software Engineering" + "\n" +
                        "Auburn University, Auburn AL 36849.  All rights reserved.";
      }


      protected override void Dispose(bool disposing)
      {
         if (disposing)
         {
            if (components != null)
            {
               components.Dispose();
            }
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code
      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
```

```
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(AboutForm));
            this.buttonAboutOK = new System.Windows.Forms.Button();
            this.groupAbout = new System.Windows.Forms.GroupBox();
            this.pictureBoxAbout = new System.Windows.Forms.PictureBox();
            this.linkLabelAbout = new System.Windows.Forms.LinkLabel();
            this.labelAbout = new System.Windows.Forms.Label();
            this.groupAbout.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxAbout)).BeginInit();
            this.SuspendLayout();
            //
            // buttonAboutOK
            //
            this.buttonAboutOK.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.buttonAboutOK.Location = new System.Drawing.Point(360, 296);
            this.buttonAboutOK.Name = "buttonAboutOK";
            this.buttonAboutOK.Size = new System.Drawing.Size(75, 23);
            this.buttonAboutOK.TabIndex = 0;
            this.buttonAboutOK.Text = "OK";
            this.buttonAboutOK.Click += new System.EventHandler(this.buttonAboutOK_Click);
            //
            // groupAbout
            //
            this.groupAbout.Controls.Add(this.pictureBoxAbout);
            this.groupAbout.Controls.Add(this.linkLabelAbout);
            this.groupAbout.Controls.Add(this.labelAbout);
            this.groupAbout.Location = new System.Drawing.Point(8, 16);
            this.groupAbout.Name = "groupAbout";
            this.groupAbout.Size = new System.Drawing.Size(704, 264);
            this.groupAbout.TabIndex = 1;
            this.groupAbout.TabStop = false;
            //
            // pictureBoxAbout
            //
            this.pictureBoxAbout.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Center;
            this.pictureBoxAbout.Image =
((System.Drawing.Image)(resources.GetObject("pictureBoxAbout.Image")));
            this.pictureBoxAbout.Location = new System.Drawing.Point(30, 32);
            this.pictureBoxAbout.Name = "pictureBoxAbout";
            this.pictureBoxAbout.Size = new System.Drawing.Size(100, 150);
            this.pictureBoxAbout.TabIndex = 2;
            this.pictureBoxAbout.TabStop = false;
            //
            // linkLabelAbout
            //
            this.linkLabelAbout.Location = new System.Drawing.Point(338, 176);
            this.linkLabelAbout.Name = "linkLabelAbout";
            this.linkLabelAbout.Size = new System.Drawing.Size(225, 48);
            this.linkLabelAbout.TabIndex = 2;
            this.linkLabelAbout.TabStop = true;
```

```
            this.linkLabelAbout.Text = "http://www.eng.auburn.edu/users/agarwra/";
            this.linkLabelAbout.LinkClicked += new
System.Windows.Forms.LinkLabelLinkClickedEventHandler(this.linkLabelAbout_LinkClicked);
            //
            // labelAbout
            //
            this.labelAbout.Font = new System.Drawing.Font("Tahoma", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.labelAbout.Location = new System.Drawing.Point(208, 32);
            this.labelAbout.Name = "labelAbout";
            this.labelAbout.Size = new System.Drawing.Size(488, 144);
            this.labelAbout.TabIndex = 1;
            this.labelAbout.Text = "Project + Author Details";
            this.labelAbout.TextAlign = System.Drawing.ContentAlignment.TopCenter;
            //
            // AboutForm
            //
            this.AcceptButton = this.buttonAboutOK;
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.BackColor = System.Drawing.SystemColors.ControlLight;
            this.CancelButton = this.buttonAboutOK;
            this.ClientSize = new System.Drawing.Size(722, 336);
            this.Controls.Add(this.groupAbout);
            this.Controls.Add(this.buttonAboutOK);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "AboutForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "<App.Title> - About";
            this.groupAbout.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.pictureBoxAbout)).EndInit();
            this.ResumeLayout(false);

        }
        #endregion

        private void linkLabelAbout_LinkClicked(object sender,
System.Windows.Forms.LinkLabelLinkClickedEventArgs e)
        {
            linkLabelAbout.LinkVisited = true;
            System.Diagnostics.Process.Start("http://www.eng.auburn.edu/users/agarwra/");
        }

        private void buttonAboutOK_Click(object sender, System.EventArgs e)
        {
            this.Close();
        }
    }
}
```