2013

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

PeTE:

Programming education Teaching Environment

Paul Martin

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management
University of North Carolina Wilmington
2013

Approved by

Advisory Committee

_____	_____
Dr. Douglas Kline, Chair	Dr. Tom Janicki

_____
Dr. Curry Guinn

Accepted By

_____
Dean, Graduate School

Contents

1 Motivation

1.1 High level overview

Teaching software programming is challenging to both students and instructors. The basic process of assigning, completing, and grading projects is filled with non-value-added tasks such as file management, compression, delivery, decompression, and navigation through folder structures. As a result, instructors tend to assign fewer projects, provide less feedback, and generally spend too much time on activities that don't promote learning.

Professional programming involves collaborative software development and professional practices that are not well-addressed in university curriculums. Tools such version control, task tracking, bug tracking, and workflow management systems are generally not covered in classes. As a result, graduates are not well-prepared to use these tools, they must be trained by their employers, and they must adopt professional software development practices after graduation.

Professional software development tools and systems were mainly developed to aid software development teams but can also be useful in an educational context. They can not only relieve the burden of non-value-added tasks, but would also expose students to additional professional work habits.

This project used Microsoft's Team Foundation Server (TFS) to teach .Net programming in the Information Systems and Operations Management department at University of North Carolina Wilmington. Although the project focused on .Net programming, the concept applies to programming education in general.

Specifically, the project involved:

- Setting up a TFS system for all .Net programming classes
- Creating workflows for student project assignment, submission, feedback, grading.
- Creating user guides for students and Instructors to use TFS.

The basic custom workflow for programming assignments is, (refer to Appendix A):

- Instructor assigns projects as a Work Item Task (WIT) in TFS
- Student retrieves the WIT from (Visual Studio, VS) connected to the TFS server
- Student programs and checks in their code regularly. On each check in they have to associate the check in with the WIT task for the assignment.
- If the student needs help, the WIT is assigned to the instructor marked as "Need Help" with student comments in the WIT
- The Instructor opens the WIT and provides comments, or makes minor changes or comments in the code.
- The Instructor then reassigns the WIT marked as "Feedback provided" back to the student.

- When the student is ready for grading, they send the WIT to the Instructor marked as “Ready for Grading”.
- From the Team project level, the Instructor performs a Get Latest Version from source control which gets all of the students’ most recent versions of their projects.
- The Instructor opens the student’s WIT and their solution file for grading. Comments can be added directly in code and checked in, and there is also an additional space on the WIT for entering a grade and comments.
- The Instructor then completes the WIT which the student can immediately retrieve from their Graded Assignments query in VS.
- Once all WITs have been graded, a progress report is done for all students for that assignment and exported to Excel. This contains student name, assignment name, due date, student submission time, grade, comments, Instructor or TA who graded the assignment, the grader’s submission time, and the description which contains any back and forth communication between the Instructor and student.

For this workflow, all work is version controlled. Each changeset, which is a new version of the current files created by source control, is tracked by associating each check in with the students WIT. This allows for more detailed information in the history of the WIT and prevents students from making changes and checking them in after they have submitted their WIT for grading. It also allows for more detailed reporting using SQL Reporting Services. While this workflow does require some one time work at the start of class from the Instructor, it makes file management easier for the Instructor and student.

1.1.1 Central repository for code

Using TFS, a central repository was created for all students .NET programs. By having a central code repository, this eliminated the need for students and instructors to spend time on non-value-added activities including file compression and decompression. It also eliminated the need for students and instructors to exchange files. The central repository gave the instructor access to every student’s workspace which aided in providing assistance outside of class and grading. It also allowed for students to do all of their work in one location and provided file management for the students’ files.

1.1.2 Streamlined submission, grading, and feedback

Student submission, grading, and feedback are areas in the .NET programming education where substantial time must be spent on non-value-added activities. With the use of TFS, these activities were streamlined. The students’ submitted their work to the central source, and instructors had direct access to this eliminating the need of file exchange between the students and instructors. By removing the need of file exchange, the grading process was easier on the instructor and saved them time. This allowed for the possibility of more assignments and multiple iterations of an assignment. It also allowed the instructor to be able to provide better, more substantial feedback to the students.

1.1.3 Integration with IDEs

While TFS was the backbone for this system, the students benefited from the fact that it integrates with the Visual Studio Integrated Development Environment. From Visual Studio (VS), the student could connect to the TFS server, and get to their workspace and WITs. The students did not have to understand the detailed specifics of the server, only that the TFS server is where their work was located. VS made it easy to connect to the TFS server, and did not clutter the students' workspaces with additional tools. Everything dealing with TFS from VS was done from the Team Explorer tab which is located next to the Solution Explorer tab.

1.1.4 Professional practices and tools

Graduating students are not lacking knowledge in .NET programming; they are just inexperienced with collaborative development environments. With the use of TFS, we were able better utilize VS to incorporate a collaborative development environment in the .NET classes. Instructors concern themselves with introducing students to as many professional practices and tools possible. With integrating VS with TFS, instructors had more resources available to them to provide to students.

TFS also introduced students to a wide variety of professional tools that are widely used in the industry. Introducing them to the concepts of these tools and utilizing them throughout the .NET programming education improved their overall knowledge, and better prepared them to become professional programmers.

1.2 Improved student learning

One of the major benefits of TFS is that it helped students learn better. Many of the non-value-added activities were eliminated with the use of TFS allowing for more time to be focused on learning. With student submissions, grading, and feedback being streamlined, the quality and number of assignments that could have been given were increased. Being able to have an assignment that had several iterations that build upon each other would enforce an iterative development approach, and promote the improvement of student programs through instructor feedback. It also promoted a learning environment in which getting outside help from the instructor is easier. TFS allowed for peer review of programs, which would help students learn the essential skill of being able to trace another programmer's code.

1.3 Student and instructor time savings

Incorporating PeTE into the class saved student and instructor time. Streamlining the teaching of .NET classes reduced time that was used on activities that did not directly contribute to learning .NET programming. That time was able to be used to focus on activities that directly contribute learning .NET programming. Instead of file exchange between the Instructor and students, the students could focus on learning the concepts of WIT, version control, and the iterative approach to development. The goal of streamlining also provided Instructors with time saved to be able to provide more feedback and assist students easier. Typically, Instructors would like to be able to provide more substantial feedback to their students and be able to assist them but are restricted by the challenges of teaching .NET programming.

2 Background

2.1 Learning theory

Learning is an inherently complex process. Commonly defined, "learning is a process that brings together cognitive, emotional, and environmental influences and experiences for acquiring, enhancing, or making changes in one's knowledge, skills, values, and world views (Illeris, 2000; Ormorod, 1995)."

Observing the current learning theory, a better understanding of what affects the success of students in programming education was able to be reached. This theory proposes that there must be a substantial number of challenging assignments. It also proposes that reducing the amount of time it takes to return feedback to students will increase student knowledge and retention.

In Gagne's work on the Conditions of Learning (The Conditions of Learning, Robert M. Gagne), he observed the failure of student learning was a result of the gaps in knowledge needed to perform the related sub tasks of a given task. From this he concluded that instruction should be composed of a set of tasks that are introduced in sequence ensuring that the student masters each task and be able to transfer this knowledge to complete the final task. In Gagne's work on the conditions of learning, he identifies nine events that must occur for learning to be successful.

1. Gain Attention: Perception
2. Inform objectives: Expectancy
3. Stimulate recall of prior knowledge: Retrieval
4. Present stimulus material: Stimulus
5. Provide learner guidance: Encoding
6. Elicit performance: Response
7. Provide feedback: Reinforcement
8. Asses performance: Retrieval
9. Enhance retention and transfer: Generalization

He also identifies instructional events which alone do not produce learning, but instead support the student's internal process. There are two instructional events that should be observed, the first being preparing for learning. This includes the events gain attention, inform objects, and stimulate recall of prior knowledge. In order for learning to occur, the instructor must first capture the attention of the students. Once the instructor has the attention of the students, they need to inform the students of the objectives for the lesson. This will give the students a degree of expectancy, and helps motivate the student to complete the lesson. Finally, any opportunity to associate new information with prior knowledge should be taken as it facilitates learning.

Another important instructional event to observe is acquisition and performance. This involves four of the Gagne's nine events including presenting the material the students in meaningful sections. Since there are different learning modalities, this material should be presented in a variety of different ways which can include text, graphics, audio, and video. With the presentation of new material, the instructor must provide learning guidance. This can be in the form of examples, graphical representations, and case studies. The student then must be able to practice the new skill. Being able

to practice the new skill helps confirm understanding, and repetition through practice increases the probability that the information will be retained. Finally, students need to be provided with immediate feedback of their performance. The quicker a student receives feedback, the more likely it is that they will learn.

2.1.1 Instructors limit feedback

Providing substantial feedback is one of the most challenging areas for an instructor teaching a .NET programming class. With increasing class sizes and a limited number of instructors, it makes providing feedback increasingly difficult. To further complicate the situation, the feedback that is given to the students is not applied to their work. They see what they had points taken off for, but no opportunity is available for students to use the feedback to fix the mistakes they made. With the class sizes, and the complexity of grading and providing feedback, instructors tend to limit the substance of their feedback. If students were encouraged or required to use the feedback to fix the mistakes they made, this could be viewed as practice. Students would be able to confirm what they know, learn what they missed, and retain this information. With students using the feedback in a meaningful way, it would provide more incentive to the instructor to provide more substantial feedback.

2.1.2 Instructors give delayed feedback

With .NET programming classes, delayed feedback on assignments is a normal occurrence. Once an assignment is submitted by the class, on average it takes at least a week to get results back to the students according to the interview with Dr. Douglas Kline. (Appendix M.2) Reinforcement is one of Gagne's nine events (The Conditions of Learning, Robert M. Gagne). It states that students need to receive immediate feedback in order to learn. Currently, feedback is provided to students much in the same way that students receive feedback on tests and quizzes. Gagne does distinguish between these two types of feedback. The feedback on tests and quizzes is meant to provide scoring. It is an evaluation of the knowledge that the student has obtained. For lessons, the feedback is meant to be used for comprehension and encoding purposes. Therefore, reducing the amount of time that it takes to return feedback to students would increase the learning process.

2.1.3 Instructors limit the number of assignments

Due to the complexity of .NET programming classes and the size of the classes, it is common for instructors to limit the number of assignments. This is done for several different reasons. The first being that the instructor has to balance the amount of assignments with the number of students and the amount of assistance the instructor is receiving. With too many assignments, it can become impossible for the instructor to help every student that needs it. Instructors also limit the number of assignments because the time consuming nature of the grading process. If the instructor assigns too much, they are in danger of being unable to grade the assignments as well as providing much less feedback. This is contradictory to the learning theory in that it states that there must be a number of challenging assignments for learning to occur. However, with increasing class sizes, limited number of instructors, and the current teaching process, increasing the number of assignments is not possible.

2.1.4 Assignments are restricted to one iteration

With the size of classes, limited number of instructors, and the complexity of submission, grading and feedback; assignments are completed in one iteration. An assignment is given and completed from start to finish. This leaves the delayed instructor feedback not being utilized to have the students use the feedback to fix their mistakes. Fixing their mistakes would provide additional practice, confirm what the student knows, learn what they missed, and help them retain this information. This approach to assignments does not reflect the iterative nature of software development. Code is written for a program, and then continually revised until the final product is produced. Structuring the assignments where the overall problem is broken down into iterations would help teach this iterative approach. This would also allow for instructors to provide feedback and have the students use this to fix any problems with the iteration. Once the problems are fixed, they can continue with the next iteration and build upon their work. This would also follow Gagne's theory that in order for learning to occur, the problem must be broken down into manageable sections in which the students become masters of each section. This leaves them prepared with the necessary knowledge to complete the final task.

2.2 Microsoft Team Foundation Server 2010

Microsoft TFS 2010 "is the collaboration platform at the core of Microsoft's application lifecycle management (ALM) solution" (TFS 2010 Overview, Microsoft). It is a professional development environment "which enables everyone on the team to collaborate more effectively, be more agile and deliver better quality software while building and sharing institutional knowledge" (Microsoft Visual Team Foundation Server 2010 Datasheet, Microsoft). TFS consists of collections which are containers for related team projects. Team projects are the instances of a project. Each TFS team project uses a process template to define the software development methodology being used. In addition, each team project contains source control, work items, enforcement policies, builds, and reports.

2.3 Source control

It became apparent as early as the late 1970's that source control was needed in Software Development. Today, the majority of Software developer's agree that "source control is fundamental to the practice of modern software development" (Joel Spolsky). Source control, also known as version control or revision control, and it works with a centralized server for a project's data. It allows developers to connect to the server and get the latest version of a file and check the file back in after they are done making any changes. When a file is added to the source control it is marked as being version 1. Anytime the file is changed and added back to source control a new version using incremental numbers is created. The main concept of source control is to allow developers to work on the same code with new versions of each change to file to be created. This is done in an effort to keep developers from having to redo work due to lost or overwritten files.

Source Safe was one of the first major systems that were being used. It uses VSS 2005 and was adopted by Microsoft at that time. There are many criticisms of Source Safe one of them being that "it uses a direct file-based access mechanism that allows any client to modify a file in the repository after locking it. If a client machines crashes in the middle of updating a file, it can corrupt that file. "(Visual

SourceSafe: Microsoft's Source Destruction System, Alan D. Smet)" Source Safe also allows you to check in unsupported files to the source control without any warnings. If SourceSafe does not support the file, it corrupts it and you are unable to remove the corrupted file. This further complicates the development process when it comes to builds. Before a final installation build can be done, the corrupted files must be identified and removed by hand before the installation build can occur. Basically, SourceSafe gives the illusion of safety and control, while exposing your project to risk. It also teaches developers bad habits like avoid branching, exclusive locks, easy permanent deletions (Coding Horror, programming and human factors, Jeff Atwood).

It is possibly that even though source control is fundamental in Software Development, the reason that start up's and small companies are hesitant to adopt new software for source control is due to their early experiences. Any developers who worked with one of the early bad systems with their many problems are skeptical that the new systems available are any better or that they will just cause different or new problems. If you refer to Appendix B, it shows the different Software available. Taking a look at SourceSafe it clearly illustrates the problems it had.

2.4 Programming education

Professional programming requires knowledge of the fundamentals of .NET programming, and the use of the collaborative development environments. All of the source code for a program is stored in a central location. Work is typically assigned using work items, and bugs are tracked and fixed using bug tracking items. Knowing how to program in .NET alone is not sufficient for being able to be a professional programmer. Students must also learn the concept of these collaborative development environments, and how to use them. Currently, this training is left to the company that hires the student. Before the student can be productive, the company must spend time and money to train them to work in their development environment. Using TFS, we can prepare students with the knowledge to be able to be productive from the day they are hired. This would give students a competitive edge against other prospective job candidates who lack this knowledge of professional development environments.

3 Microsoft Team Foundation Server 2010

3.1 Background and specifications

Microsoft TFS 2010 "is the collaborative platform at the core of Microsoft's application lifecycle management (ALM) solution" (TFS 2010 Overview, Microsoft). It is a professional development environment "which enables everyone on the team to collaborate more effectively, be more agile and deliver better quality software while building and sharing institutional knowledge" (Microsoft Visual Studio Team Foundation Server 2010 Datasheet, Microsoft). TFS consists of collections which are containers for related team projects. Team projects are the instances of a project. Each TFS team project uses a process template to define the software development methodology being used. In addition, each team project contains source control, work items, enforcement policies, builds, and reports.

3.2 Centralized source code database

TFS includes a central repository for development teams which is integrated with VS10 (Visual Studio 2010). This is an essential part of a development environment since there are typically several to hundreds of developers working on the same source code. With a central repository you can ensure that all developers on a team are working on the same source. However, having a central repository for source code is not enough. There are the issues of developers having to redo work due to another developer unknowingly overwriting their work and collisions occurring when multiple developers are working on the same code. TFS server uses version control, changesets, and collision detections tools to prevent these events from happening.

3.2.1 Version control

TFS 2010 has come a long way since the early days of source control. It works like other software where there is a central repository for the source. The integrity of the source is controlled through check-ins with work items and check in policies. When a project is started it is labeled changeset 1. Every time that a change is made to the current version, a changeset is created. These changesets allow for you to view the history of a specific version and the changes that were made which are shown by the changesets. TFS's source control also supports atomic commits which protects the repository. "If an operation on the repository is interrupted in the middle, the repository will not be left in an inconsistent state" (Version Control System Comparison, Shlomi Fish). It also provides check in polices to control the integrity of the source. This will be covered in more detail under the work items section.

Another way that TFS's source control protects the integrity of the source is through collision detection tools. If two developers are working on the same file and make changes, the first developer who checks back in the file will have a new changeset created. When the second developer goes to check in the same file, TFS looks for any collisions that have occurred and identifies them as conflicts. Both developers are notified of the conflicts and are given options on how to resolve the conflicts. If TFS does not detect any collisions then it will auto merge the two files together and create a new changeset.

3.2.2 Versions and changesets

In TFS's source control, when you check a project in for the first time TFS creates a baseline version of your project. Versions should be considered as milestones for a project where you have your baseline version, then version 1, 2, and so on. Anytime that a change is made to a version a changeset is created for the changed files. This allows tracking all of the changes made to a current version which has many benefits. By viewing the history of a file, you view the changesets that are associated to it. It also allows you to view the changesets as files as they were in that changeset and compare it to other changeset. This allows for reporting to show the specific changes that were made to a specific version.

3.2.3 Branching, merging, and shelving

TFS offers some additions to version control including branching, merging, and shelving. A project will have a source code base known as the trunk. Small tasks can often be done directly in the source because the small impact on the overall system they have. However, large changes with widespread impact need to be carefully researched and planned. TFS's version control allows for branches to be

created off the trunk. This allows for large changes to be carefully executed before they are merged into the source. With branches, there is a full view into the history of the branch hierarchy. Creating branches is also useful for separating different version releases from the working source code. When a version is marked for release, a branch can be created off the source for that version and the source will be continued to be worked on. If a customer reports a bug in version two of the software, the bug is fixed in the branch not in the source. This allows for an update to be released for that version of the software. When the next version is marked for release, the bugs that were fixed in the branch of a previous version can be merged into the source.

Merging is used for several different reasons. The first reason is when a branch is created to work on a major task or when a branch is created for a completed version of a project. In the first case where a major task is be worked on and considered, if it is decided that it should be part of the trunk, TFS will allow for the branch to be merged into the source. For the second case with a branch being created for a completed version, it allows for the new changes to be merged into the new version. Merging is also a useful tool when two developers are working on the same code. When the developers check code in, TFS checks for any conflicts between the code. If no conflicts are identified, TFS will automatically merge the two developers' code together. If there are conflicts between the two developers code, it will notify both developers of the conflicts and let them resolve the conflicts before merging the two developers' code.

Another benefit provided by TFS for merging is allowing you to easily see your branches and the changesets associated with it (Refer to Appendix C and D). TFS shows you your branches in a hierarchy and allows you to view the changesets associated with each of the branches. This gives you better control over your branches and provides more information as to what is taking place in each of these branches.

TFS also provides a shelving feature. If you are working on a file and are not ready to check it into the trunk, you can choose to shelve the file which saves it into the source control, but not into the trunk. This is useful if you are using check in policies. For example, you can set a check in policy where the developer must before a build on the files before checking it in. If the build fails, it will shelve the file until a successful build is run at which time it will merge it back into the trunk.

3.3 Collections and team projects

TFS uses collections as containers for related team projects. When a new collection is created, TFS creates a single database that stores all of the data for the team projects in that collection. The benefits of collections are first, that you can isolate non related projects from each other non related projects. The second major benefit of collections is that all team projects contained in the collection can share reports, work items, process guidance, as well as code base. While team projects in a collection can share all of these items, non related projects in another collection are completely isolated from each other.

Team projects are what would be considered individual projects. When you create a team project you choose which process template to use for the team project which is determined by your software

development methodology. Each team project has its own source control which can be shared with other team projects in the collection. You can create groups at the collection level to give access to all team projects, or you can specify groups at the team project level if you have different developers working on different team projects. Also included are work items which can be tracked by a team project, or you can specify different areas and iterations to provide more detailed tracking. There are also reporting and builds included in each team project.

3.3.1 Process templates

When you create a new project, you have to specify the process template that you are going to use. TFS has two out of the box templates that you can use which are recommended. A process template is defined as,

> "A process template is a collection of files that together define various process elements of a team project in Team Foundation Server."(msdn website)

The process template includes a series of folders and files that are all part of the process template. There is also an xml file which contains four pieces of information about your process: the name, the description, the plug-ins required to create a project, and the xml file associated with each of the plug-ins. (ALM p.617)

A process template defines key aspects of a team project that affect how a team works. You can customize a process template, in which you can define the initial security configuration for team project, new work item types and queries, reports for monitoring and status, and the iterations and organization units that are used. The process templates define the initial process settings for team projects. However, you can customize these process settings after a team project has been created. There are some exceptions, the main one being the test resolution states that are defined for Microsoft Test manager. (msdn http://msdn.microsoft.com/en-us/library/ms243782.aspx)

To fully understand process templates and how the work with TFS, let's first look at the process. A process is a method or system used by teams while developing software applications. (ALM p.549) There are many well defined methods such as the waterfall approach. These methods require for team members to comprehend the value that the process offers, and to follow the steps or procedures laid out by those processes. ALM authors point out that many times these heavily documented procedures are put in large binders, given to the development team, and are largely untouched. They cite the book Software Engineering with Microsoft Visual Studio Team System by saying that these processes are untouched because most of these processes and tools force a one-size fits-all approach and do not factor in the varying needs of team members. With the process templates, TFS allows you to customize the environment for your team members and to choose and adopt processes that match the needs of the team. They integrate the process rules into the environment providing a much higher probability that these processes will be adopted by the team. In short, the process templates provide a way to introduce the defined processes to the team without hindering their work.

Every software development company uses some form, variation, or combination of some Software development methodology. Another benefit of the process templates is that they allow development teams to more closely follow their desired development methodology. For example, companies using the Agile development approach often get stuck in the middle between Agile and Iterative. The Agile process template for TFS sets up the team project with the framework for this approach and assists companies in moving away from this mixed approach and fully adopting the agile approach. This is achieved with the process templates and the use of the iterations and the concept of creating branches for each version. With a branch being created for each version, these companies can move away from the iterative approach where they work in the trunk to fix bugs and issues, but instead use the branches of the versions to fix these problems.

3.3.2 Areas

The area classification in TFS is a logical division of a team project. This allows for better tracking through the use of the existing workflows. Depending on the type of project, areas can be defined as Database, UI, Components, and Test. Work Items can be assigned to a specific area to provide better reporting and management. Ultimately, areas provide a more granular view of the state of a project.

3.3.3 Iterations

Iterations are a standard part of the Agile process template. Through the use of iterations an organization can view detailed reports for each Iteration. The use of Iterations is not limited to the Agile development approach. They can be used as milestones for a particular project for a development team that does not follow the Agile methodology and still benefit from the detailed reporting from each milestone.

The iteration classification in TFS is a chronological breakdown on releases and development iterations. Typically you would define a release and then have x number of iterations to complete the release. The release would contain the user stories that were defined for a release. These illustrate the functionality that will be added. For each release, you typically have about seven iterations depending on the methodology being used. Iterations can be a week or four or more weeks. From experience, I have noticed that short iterations are more useful. The longer the iteration, the harder it is to define what work can be done. With an iteration lasting three weeks, it is much more manageable to define what needs to be done and to accomplish these tasks.

3.4 Work items

TFS provides six out of the box standard work items including Bugs, Shared Steps, Task, Test Case, User Story, and Issues. These work items are not the only types that available, you can create new types as needed. Each work item has fields, a layout, and a workflow associated with it. Work items are important because they help you control your development, testing, and quality assurance processes. (msdn, Work Items) While these work items come standard out of the box, they are all customizable.

Work items are automated and help you control and track the progress of development. They are assigned to a team member, who performs the required task and completes their part of the work item. From the standard out of the box work item, this helps in identifying what team members did what. However, these can easily be improved by customizing the workflows of the work items.

3.4.1 Fields

Part of a work item is its fields which

> "display data, or let users input data or selection options" (msdn, Work Items).

Each field has a name and a field type which can be a string, integer, double, date time, plaintext, HTML, history, and treepath (Appendix H). You can also specify the reporting attribute to be a detail, dimension, measure, or none based on your reporting needs. Setting the attribute to be a dimension adds the data in the field to both the database and the SQL Server Analysis Services cube (msdn). This will increase the size of the cube significantly. If you just need reporting and not SQL Server Analysis Services it is best to set it to detail. This adds the data to the database and allows you to still run reports.

With each field in a work item you can specify rules. There are nineteen rules that are predefined by TFS and you can determine which to use and how to use them. They range from valid values that are allowed in the field to read only. For example, one of the biggest headaches with work items is the "assigned to" field. By default most users when they create a new work item and click the drop down box for assigned to will see a list of users and computers that are related to their team. To fix this you can set a rule on the "assigned to" field for valid values. You can mark the box exclude groups and create a new list including any TFS groups related to that project. Now when the user goes back to create a work item, they will only see the users they included in valid values in the assigned to field.

3.4.2 Layout

Each work item has a layout of the work item form (Appendix H). The layout is fully customizable to show the information that is needed, exclude any unnecessary parts, and include any fields that you created in a control. The layout works by using groups and tab groups. Groups can contain other groups, columns which separate the groups on the form, and controls which contain fields. There are also tab groups which show up as tabs on the form. Inside a tab group TFS allows you to create tab pages which have groups, columns, and controls.

3.4.3 Workflow

Each work item has a standard workflow associated to it (Appendix I). To better fit these to meet the needs of an organization; you can customize every detail of the workflows. Microsoft recently released the Power Tools feature which is a plug-in for VS10 and TFS. In the past, customization of workflows had to be done through modifying the XML. The Power Tools plug-in provides a GUI interface for customizing the work items. Each work item has the fields that are associated with it, the layout of the work item form, and the workflow itself which is viewed as a state diagram (Refer to Appendix H for

Fields and Layout). Viewing the workflow of a work item as a state diagram they have transitions and states. A transition is the event that is triggered to move the work item into a new state. With each transition there is also a reason associated with why the work item is moved into a new state. Defining clear reasons through the work item workflow an organization can benefit from more detailed reporting.

The way the workflow for a standard work item task is that it is assigned to a team member (Appendix E). When the team member gets the work item task it moves into the state of Active. When they are done with the task, they mark it as complete. This does not meet the needs of most development companies. However, with the use of the Power Tools, you can create new fields, modify the layout of the form, and add new states and transitions in the workflow. For example, you can have better quality control by adding an additional state to the work item task (Appendix I). In this example situation, I want to ensure that the task was completed and that it meets the needs of the requirements. To do this, an additional state can be added where when the original team member assigned to the task is complete; it is sent another team member who performs quality control. The reason that the original team member would select would be "Ready for Review". If the quality control team member determines that the task does not meet the requirements, they can send the work item task back to the original team member with comments in the work item about what needs to be done and would mark the reason as being "Incomplete". When the quality control team member final signs off with the reason being "Complete", there are now two team members who have signed off on the task giving the organization more control over the quality of the product.

Workflows allow for additional states, transitions, and reasons to be included in the work item which provides a more structured environment and more detailed reporting. Additional fields can also be added which can provide additional information in reports. Some information can be obtained through the transitions and states of the work item workflow, but some may require you to add additional control to the work item form. While the work item form is part of the work item itself, the information is part of the workflow. With the Power Tools plug-in Microsoft has made it easy to create custom fields and add controls on the work item forms for those fields giving an organization full control over their environment.

3.4.4 Enforcement policies

Often there are many processes that you want to enforce but depending on the development environment, it is left to the developer to follow. TFS allows for these policies to be enforced directly in the environment. There are check in policies available which include requiring a check in to be associated with a work item, specifying which work items are valid to associate to, enforce that comments are added at each check in, and enforce that the build succeeds before a file can be checked in. You can also require that code analysis is run and any tests are successfully before checking in. These are just a few of the policies that you can enforce. The major benefit is that it is no longer left up to the developer to decide if he wants to follow these policies. During a check in, if one of the defined policies is not satisfied the developer will receive an error with a list of the policies not met for check in.

This ensures that your policies are being enforced using work items with little to no direct involvement from management.

3.5 Testing

Testing is a very important aspect of software development. Good practice is to test at every step including unit testing, component testing, integration testing, system testing, and user testing. TFS has a powerful addition which is the Lab Management. It works with Hyper-V, and allows you to create virtual environments that are isolated on their own domain. You can perform cross platform testing by creating virtual machines with the platforms you need to test. When the tests are run, they are run on all platforms and generate detailed reports. TFS also has test assemblies which can be used to specify unit tests to be run when you define the build definitions.

3.5.1 Unit tests

When you perform a build in TFS, it allows you to define which tests you want to run against the code. The tests are organized into test assemblies and you can choose which assemblies to use during the build. In TFS, you define which assemblies to use in the build definition. Every time that TFS runs the build, it will run it against your tests and produce a report. If you require a build after each check in, it helps ensure that any changes did not break the existing unit tests. This helps ensure that the continually changes made to the source do not break existing unit tests. You can also specify that if any of the tests fail, that the changes will be shelved until they are fixed. This will be covered in more detail in the Enforcement policies section.

3.6 Builds

TFS comes with Team Foundation Build which must have at least one build machine. The build machine can be a physical or a virtual machine. Each team project collection has a dedicated build controller dedicated to it. This build controller accepts any build requests from any team project in the specified collection. The build machine houses the build agents which is dedicated and controlled by the build controller dedicated to the team project collection. The build agent is responsible for doing the processor-intensive and disk intensive work which includes getting files from and checking files into version control, provisioning the workspace, compiling the code and running the tests. TFS's build system is scalable where you can start with just a few agents and scale your system out by adding more build agents.

With each team project in TFS, it allows the ability to create build definitions. Build definitions define the location of the project in the source that you want to build, the build template, the files that you want to build, and any test assemblies that you want to include in the build. Build templates are customizable and define how the build will occur. TFS comes with three out of the box templates which are customizable in a GUI interface, and it gives you the ability to create new build templates to meet the needs of the development company.

3.7 Configuration management

Software development configuration management is responsible for tracking and controlling changes in software "(Software Configuration Management Strategies and IBM Rational ClearCase: A Practical Introduction, 2nd Edition. David E. Bellagio, Tom J. Milligan)". The tasks that are included are identifying configurations, configuration items, and baselines. It also includes implementing a change process, recording and reporting, configuration auditing, build management, process management, and environment management. Once a piece of software is developed, its initial version could be identified as the baseline. As changes are made to the baseline, these are identified as configurations. This allows for multiple versions to be clearly identified with the ability to compare to the baseline to observe the specific changes.

TFS is an enterprise level configuration management tool. The process templates allow for the overall software development methodology to be defined and controlled. Using the Agile process template as an example, it helps identify configurations or versions using the branching method discussed in the Branching, merging, shelving section. The use of iterations and work items makes it easier to determine which items are involved with each version. In addition, work items and changesets allow for reporting to show what items were changed and how in each version. Build management is handled with build definitions, process management is controlled by the work flows of the work items, and the environment is managed using the process templates. With all of these aspects tied into the development environment TFS provides powerful reporting and configuration management tools.

4 Programming education

4.1 Overview

Programming education is very challenging to both instructors and students due to a limited number of instructors and increasing class sizes. At the introductory level instructors have to teach highly abstract concepts and develop the students' problem solving skills. With the increasing class sizes and the varied backgrounds of students it becomes increasingly difficult for an instructor to structure a class to meet the needs of all the students. Due to the challenges that instructors face, instructors are often left trying to cover the basic syntax that students need causing limited time to develop their problem solving skills (On the Development of a Programming Teaching Tool, Journal of Information Technology). It is typically the case that students do not have assignments which involve collaborative work or peer review. These are also skills that students need to develop but the inherent challenges of programming education often hinder these skills from being developed.

4.2 As-is

Refer to Appendixes J.1 through J.5 for swinlanes of the current processes. There is no standard process in the ISOM department for .NET programming education. A formal system for a central location for students' work is also absence. However, there is one exception which is Dr. Janicki's MIS 413 class. He has created a virtual server miscapstone, where students' can copy their code from Visual Studio 2010 and both students and the instructor have direct access to it.

With no formal system, all student work is saved either on local hard drives, jump drives, or network share drives. This causes extra work to pass project files between students and instructors. This makes

grading and providing help to students outside class problematic. Projects are assigned to students with only one iteration, and grading takes on average one to two weeks. Any feedback received from the instructor by the student is not applied to the project. The number of assignments and their complexity is also limited by increasing class sizes and limited instructors. There are currently several activities required that take substantial time that are not learning related including grading, feedback, student help, and student submission.

4.2.1 Faculty interviews

Five instructors were interviewed on the current processes used in Programming Education (Appendixes M.1 – M.4). During the interviews with the faculty in the ISOM department, it quickly became apparent that there is no current standard for any of the processes associated with these classes. Each instructor has a slightly unique way of creating and assigning projects, grading, and providing feedback. The one aspect that was the same for all instructors was the acknowledgement that a large portion of time each semester goes to non-learning related tasks. The task of getting the students projects in one location is one of the largest consumers of time. With no central repository for students work, the process of copying student files, unzipping files, and reviewing their work is cumbersome and time consuming.

Another issue that is experienced by all instructors was difficulty in providing substantial feedback in a reasonable amount of time. The current grading process is so cumbersome, that it takes on average one to two weeks to get projects graded and returned to the students. It is also a very linear process which does not support incremental work on projects. The projects are completed, graded, and the feedback received is not used to improve their original work. Due to the lengthy nature of the grading process, encouraging incremental work on projects is not possible.

The number and complexity of assignments was also addressed in the interviews. Instructors made it very clear that due to the complex nature of providing help to students outside class and the lengthy process of grading, assignments had to be limited. Additional assignments are out of the question as professors do not have any additional time. The assignments are also limited to one iteration because trying to grade multiple iterations and provide feedback is not support by the current process.

4.2.2 As-is examples

The main problem with the current process is a lack of a central repository. This affects both the students and the Instructors. Refer to Appendixes J.1 through J.5 for the as is diagrams. For students, there is no standard across the department for the students' workflow. With students' lacking a central repository for their work it causes students to save their work on local hard drives, jump drives, and network share spaces (Appendix J.4). Student's often have multiple versions of a project that can be scattered across multiple storage devices making it difficult to keep track of the current version they are working on.

Instructors' also suffer from the lack of a central repository when it comes to grading and feedback. When students' need to submit their programs they have to create a zipped folder containing all of their project files and either upload to Entropy or e-mail to the Instructor. The Instructor then has to retrieve

all of the students' zipped folders, copy them to their desktop, manually unzip them, and either opens them up in VS08 one at a time or print out copies of their vb code for review (Appendix J.5). This is time consuming, cumbersome, and viewed as a burden on the Instructors. In order to provide feedback Instructors use printed checklist which are filled out and handed back to the students. The whole grading process can take up to two weeks before results are returned to the student's. By this time, feedback is read but not utilized to improve their work.

Dr. Janicki's current process is able to provide students with a central location for their code by using the miscapstone server (Appendix M.1). However, due to the challenges of programming education this central location is not used for students to work directly on their code. The students work locally and copy their files to the server. While this does not address the issue of students working in multiple locations, it does ease the process of submission and grading. When it is time to grade Dr. Janicki is not burdened with having to copy and unzip all of the students' files. He has the students' e-mail him a link to their project which he can then just click on and it runs on a live server. This allows him to bypass the issues of permissions being ignored when it is run locally.

4.2.3 Swimlanes

Referring to Appendix J.1, this is the high level overview of an introductory level .NET programming class. When a class starts, there is some work to perform the initial class setup, refer to appendix J.2. In the initial class setup, the instructor has to create a web page for the class and then setup the class in Entropy. When an instructor needs to create an assignment, refer to appendix J.3, they must first create the details of the assignment. The instructor then adds these details to the class website, and uploads the details to Entropy.

Once the assignment has been assigned, students begin work, refer to appendix J.4. The students begin work on the assignment, saving their work to one of their individual storage devices. If a student reaches a point where assistance is required from the instructor, typically the student will e-mail a zipped file of their work or provide a jump drive to the instructor. The instructor has to copy the files to their desktop, and then open the project in Visual Studio. Any work is saved and given back to the student in the same format they provided to the instructor. If the student's primary workspace is a portable jump drive, the issue is not as severe. However, if the student's workspace is a network shared drive or a local hard drive, there is the potential for version confusion and even loss of work. When the students copy the work back to their primary workspace, they can overwrite existing files or be forced to rename the project creating multiple versions.

Once the student has completed their assignment, they are required to upload it to Entropy or e-mail it to the professor. This generally requires that they compress their work in a zipped folder first. This can be problematic if the student does not include all of their necessary files to the compressed file. The instructor would be unaware of this error until they went through the process of copying and unzipping. This can cause the instructor to have to spend more time on the grading process by having to get the student to resubmit a complete zipped file of their assignment.

The current instructor grading process is timely and cumbersome, refer to Appendix J.5. First, the Instructor will create a grading checklist for the assignment defining the areas they are grading and the point values. Once the students have uploaded their zipped project files, the instructor must retrieve these and copy them to their desktop. The instructor then has to go through and manually unzip every file. Dr. Kline's process involves using a self-written program called ProjectPrinter which prints out the files needed to grade for each students' project. Then each printout is reviewed with feedback being written on the printouts and the grading checklist is filled out. Other instructors will open each student's project in Visual Studio. He or she will then fill out the grading checklist adding any feedback in the margins. With either scenario the grades and any comments are added to Entropy and the printed grading checklist and any printouts are returned to the students. From the time the assignment is due, it can take one to two weeks to return grades and feedback to the students.

4.2.4 Estimated amount of feedback

Typically the amount of feedback that is provided to students is limited. The majority of their feedback comes from the graded checklists. These checklists merely inform the students what functionality or requirements were missing or incomplete. Occasionally short comments are added in the margin of the grading checklist. If the instructor prints out the students' vb files, he or she will add comments on those printouts. However, since the students never go back and revise their code based off of the feedback, there is little incentive to provide more substantial feedback as it is never used.

This is different with the MIS 413 class. Dr. Janicki has structured the class so that all the work done is incremental. Feedback is given in one of three ways. The first way is when their grade is entered in Entropy Dr. Janicki will add any comments there. The second way is that Dr. Janicki will print the page, whether it is a form, code, database table, or the relationship diagram in the database. He will write comments on the page(s) and hand them back to the students. The last way is that if in the code the student is doing something that is obviously wrong or there is a much better solution, Dr. Janicki will insert pseudo code for how to do it and then comment it out. He then pushes the changed file to the student's folder on miscapstone and informs the student.

Since the work done in this MIS 413 is incremental there is a need to get the students feedback by the next class. This is due to the fact that students are required to use the feedback to fix any problems that were identified. If they fail to fix these problems then on the next submission they are penalized double for them.

4.2.5 Number of assignments

The number of assignments largely depends on the class. The introductory level classes including MIS 216 typically have about 5 programming projects. More advanced classes such as MIS 413 have three mini-projects that act as a refresher and one large team client project (which is segmented into eight checkpoints). Due to the complex and lengthy nature of the grading process, the number of assignments and their complexity must be kept to a minimum.

4.2.6 Use cases

- Instructor Use Cases
    - Create class website
    - Setup class in Entropy
    - Create assignment
    - Add assignment to class website
    - Upload assignment details to Entropy
    - Copy students zipped project folders to desktop and unzip
    - Provide feedback to student seeking help outside class
    - Print student's vb files
    - Create grading checklist
    - Grade printouts provide feedback
    - Return graded printouts and feedback to the students
- Student Use Cases
    - Download assignment from Entropy
    - Create new Visual Studio project
    - Work on assignment
    - Request help
    - Create a zipped folder of project files
    - Send zipped folder to instructor

5 Systems analysis

5.1 As-planned

5.1.1 Actor diagram

Refer to Appendix K.1 through K.6 for the as planned diagrams. In Appendix K.6, the actors included the TFS System, TFS admin, Instructor, and Student. The TFS System handles the builds which can include test assemblies with associated unit tests. It could also send notifications when events occur. The TFS admin was responsible for managing user accounts, databases, and reporting services. They could also manage collections and team projects including creating and deleting. The instructor could use the reporting services and manage the databases. They were able to create a team project for their class as well as creating areas, user groups, folders for students in source control, and set permissions. They could map their class team project to a local folder as well as creating work item templates for assignment tasks and using those to create and assign work item tasks to students. Instructors provided feedback using the work item tasks and grade by completing them.

5.1.2 Server architecture

The TFS server was hosted in uncwcsbRACK05. The host for the TFS VM used 64 bit version of Microsoft Windows Server 2003, service pack 2. It was equipped with an Intel® Xeon™ CPU 2.80 GHz 2.8 GHz with 5 GB of RAM. Microsoft Virtual Server 2005 was used to handle to VM environment.

(Refer to Appendix F) The TFS server VM was running Microsoft Windows Server 2003. The following products were installed on the VM; IIS 6.0, SQL Server 2008 R2, SQL Server Reporting and Analysis Services, and Team Foundation Server 2010. With the TFS 2010 installation the included SharePoint

v3.0 was installed. The users that connected were running Windows XP or 7 with VS10 installed. Part of VS10 was the Team Explorer which allowed the users to connect to TFS from VS10.

5.1.3 Implemented TFS environment

- VM running Windows Server 2008
- IIS
- Sql Server 2008 R2, Reporting Services, Analysis Services
- Team Foundation Server 2010
- SharePoint v3.0
- Agile Process Template
- Collection for classes
- Areas used to segment students
- Iterations used for class assignments
- Customized work item tasks for assigning work to students
- Customized workflows

5.1.4 Team Foundation Server configuration

(Refer to Appendix G) The TFS server configuration consisted of two collections, classes and production. The classes' collection contained a team project for each class named by the semester, year, course name, and course section. For example, for Kevin Matthews Summer Session 1 MIS 216 class it was named 1MMIS216001. For each team project class there were two groups that were created at the team project level, instructor and students. The instructor group had full access to the entire team project while the students had very restricted permissions to prevent them from viewing other students' work. Each team project also had an area that was created for each student in which the instructor had permissions to all, but the students could only view their area. Iterations were created for each of the assignments done in VS10 that were to be turned in. In the source control under the team project class, folders were created for each student and each student was given permission to just their folder. The instructor had full permissions under the entire class team project.

5.1.5 Workflows

The main workflow was created for assignments and assigning them to the student using a work item task. Starting with the fields, some additional fields were created to handle the workflow and provide needed information. These fields included grader as a string, student as a string, grade as a double, comments as a string, and due date as a date/time field. The layout of the form also needed to be

modified to meet the needs of an educational setting (Appendix L.2). All of the unnecessary controls were removed to simplify the layout. A control for the due date, grader, and grade and comments fields was added to the form. Rules were also created for several of the fields. For the assigned to field and grader field on the form a rule was added for VALIDVALUES. I excluded groups and for assigned to I created a drop down list with the student and instructor groups. The same thing was done for the grader field but just included instructors. This prevents other users and computers from being populated in those drop down list. The due date, grade, and comment fields were marked as read only to prevent the students from making any changes to these.

The work item task workflow (Appendix L.1) was heavily customized. It had four states including student work, instructor review, grading, and closed. The workflow started by the instructor creating a work item from a template and assigning it to the student. The assigned to field and the area was changed to the student's. When the task was in the student work state, this is when the student could see their work item by running a team query and work on it. The student could send the task to the instructor for one of two reasons. The first was the default which was ready for grading. The second was used if a student needed help. A student could submit the work item back to the instructor marked as needing assistance. The instructor could then add comments to the history of the work item or add comments directly in the student's code. When the instructor was finished providing assistance, they would change the state back to student work which reassigns the work item to the student, and check in any changes to their code. This could occur as many times as possible.

When the student was finished with an assignment, they would change the work item state to grading. This signified to the instructor that the student's assignment was ready for grading. The instructor was able to view all of the students who had submitted assignments by running a team query. For grading, the instructor filled out the grading checklist and then copied and pasted the table into the history area on the work item task form. The instructor then went to the grade and comments tab on the work item and enter the grade and any comments. These were then copied into Entropy. Once the instructor was done grading, they would change the state of the work item to closed. The student could then run a team query and view the graded assignment.

5.1.6 Use cases

- TFS Admin
    - Manage user accounts
    - Database management
    - Reporting
    - Create/Delete Collections
    - Create/Delete Team Projects
- TFS System
    - Auto Build
    - Send notifications
    - Run unit tests
- Instructor (all steps total approx. 1 hour for class of 35)

    - Create a Team Project for their class (1 time)
    - Create user groups for students (1 time)
    - Set permissions on student group (1 time)
    - Create Areas for each student (35 times)
    - Set permissions on the Areas (35 times)
    - Create separate folders in source control for students (35 times)
    - Set permissions on the students' folders (35 times)
    - Map Team Project source control to local folder (1 time)
    - Create Team Queries for students (2 times)
    - Create a series of My Queries (1 per assignment)
    - Create Work Item Template for an assignment task
    - Assign tasks to students
    - Provide feedback using the assignment tasks
    - Grade by completing assignment tasks
- Student
    - Connect to TFS
    - Map source control to local folder
    - Run Team Query to retrieve assignment task
    - Create new project
    - Check in project to source control
    - Send assignment task to Instructor for help
    - Send assignment task to Instructor for grading
    - Run Team Query to retrieve graded assignment task

5.2 Benefits

Instructors are faced with the inherent challenges of the difficulties of teaching .NET programming education. They introduce students to as many professional practices and tools that these challenges will permit, but are always look to expand on what they can provide to their students. The benefits of this project was that through the use of a professional development environment which could be integrated into the current software being used; it was able to streamline the teaching process for instructors while allowing instructors to introduce students to additional tools using TFS and professional practices using work items.

5.3 Per instructor

The first and most prevalent benefit for the instructors was having a central, organized source for all students work in a class. This eliminated the need for instructors to have to retrieve student's files from e-mails or web share folders, and then copy them to their desktop in order to grade or review. TFS aided instructors when it came time to grade by providing access to all student projects on the central server. These projects could be compiled and run directly from the server. TFS also aided instructors in reviewing students work when they needed help. Students would no longer have to e-mail their files to their instructor when they needed help. If the student had a question, the instructor could quickly and easily go to the TFS server and open their project. Instructors also no longer needed to teach students

how to perform the non-value-added activities, including how to create a compressed folder of their project files. This allocated more time for the instructor to teach students material directly related to programming.

5.3.1 How much time will it save?

The initial setup of a class required more time upfront. Once the class was setup with the proper groups, permissions, areas, iterations, and source control folders; the rest of the semester was more streamlined. The process of setting up a class took roughly one to two hours for a new user of the system. For an Instructor who has become familiar with the system, it should take no longer than an hour. If builds were performed and unit tests were written by the instructor to automate the grading process, the initial time to setup up a class would have increased substantially. The increase in time required by the instructors would be due to writing unit tests. However, the unit tests had the potential to almost completely automate the grading process which would save instructors later. The unit tests could also have provided students with immediate feedback. For the scope of this project, the time that would have been saved does not include the unit testing.

For an average class, there are roughly ten assignments per semester. By providing a central source for all of students work, this eliminated the need for instructors to retrieve all of the student's projects and copy them to their desktop in order to grade. Include in the students who send incomplete projects and the process of having to get them to send it again. For each assignment, this saved roughly one hour. This freed up ten additional hours for the instructor every semester. The goal was to reallocate this saved time from performing unnecessary tasks to performing essential tasks. If build definitions and unit testing were included, the amount of time the instructor would have to spend per student would have reduced substantially and the instructor's time would have been better spent. Overall, instructors saved roughly six to eight hours for each class. If unit tests had been included in the class setup, the six to eight hours would have been used for this. This would have streamlined the grading process, but how much is unknown.

5.3.2 More accurate?

The biggest problem with the current process was the process of file transfer. When it is left up to the student to find their project and create a zipped folder, often times the students do not zip all of their project files. This allows for students to submit an incomplete compressed folder of their project files. When this happens, the instructor will not catch the mistake until after they have gone through the process of retrieving the student's files. The instructor has to notify the student, and get them to resubmit their files causing a significant loss in time. With the central source it was no longer on the student to create a zipped folder and include their project files; they just had to ensure that they checked in their work before they sent their work item task off for grading. If files were missing, the instructor could quickly see from the central source and resend the task back to the student to correct the problem.

5.4 Per student

As with the instructors, one of the biggest benefits to students was having a central repository for their work. This reduced the complexity of having to keep track of current versions of their projects scattered across local drives, network share drives, and portable jump drives. This also eliminated the need for students to upload or e-mail their work to the instructor for grading or review. The current process also opens up the possibility that the students will submit old versions. It also allows for the possibility that they can submit incomplete project files. By using TFS, we eliminated some of these margins of error. This time saved by the student could have been used on essential tasks directly related to the programming education.

5.4.1 How much time will it save?

The time that students' would save using PeTE is difficult to measure. However it can be observed that students' can potentially be saved from having to redo their work. With all their work being in source control and changesets being created after each check in, there is less of a risk that students' will lose work causing them to spend time to redo their work. Using this process, students' can also benefit from the time saved if they do not include all of their project files in the zipped folder that is sent to the Instructor causing them to have to resend their files. It can also be observed that students' can potentially save time waiting on feedback from the Instructor in order to continue on their assignments. With the use of work items a student can quickly send the work item to the Instructor marked as "Need Help". This would allow the Instructor to more quickly see the problem, provide feedback in the code or work item, and send it back to the student so they continue on their assignment.

5.4.2 How much more feedback?

With PeTE students' were be able to send work item tasks back and forth between the Instructor until they are ready for it to be graded. This provided an opportunity for more direct feedback quicker. Once the assignment was graded, the student was able to view their grade and feedback as soon as the instructor completed the assignments work item task. If builds were included, this opened the possibility for the instructor to include unit tests in the test assemblies of the build definition. If this had been used it would provide immediate feedback to the student without the intervention of the Instructor. This would have left the Instructor available to provide more detailed feedback where needed.

5.5 Big picture

The big picture is that while we are teaching students the fundamentals of .NET programming, there is a lot more we can do to prepare them for the professional work environment. TFS helped improve the process of teaching .NET programming classes saving instructor and student time. It also taught students additional professional practices which helps enforce the building of strong work habits as well as how to use the professional tools that are available. This helps produce stronger programmers who are familiar with professional practices and tools and better prepare them to become professional programmers. It also eliminated many non-value-added activities leaving more time for students and instructors to focus on learning related activities.

5.5.1 Importance of software development education

Software development is a complex process to produce a complex system. In an effort to improve software development, processes, models, and architectures are defined. However, good programmers are the key to software development. Even with well defined architectures, processes, and models; in order to be successful you must have good programmers. This illustrates the need for a strong software development education. While teaching the fundamentals of .NET programming are an important aspect of the learning process, this needs to be expanded to include the skills of a professional programmer. This involves working in a shared programming space adhering to professional practices. In order to prepare students to enter the professional workplace, they have to be educated on the fundamentals of .NET programming as well as the environments required to work in, and the type of processes that are required from a business perspective.

5.5.2 Job preparation

Looking forward, the demands for .NET programmers can only increase. It will become harder to obtain a job fresh out of college with having just the fundamentals. Companies will be looking more for programmers that have experience working in a professional environment. By hiring somebody who is already familiar with the work environment, this will reduce the amount of time and money needed to train the new programmer to be able to use the environment. A student who is familiar programming in a TFS environment, will be viewed with higher priority than a student who just knows the fundamentals giving them a competitive edge in their profession.

6 Project plan

6.1 Activities

- Setup Virtual Machine to run TFS
- Install Windows Server 2003
- Install IIS
- Install SQL Server 2008 R2
- Install TFS and SharePoint v3.0
- Configure TFS server
- Perform analysis on current processes
- Define workflows for proposed processes
- Document installation
- Document how to use the system
- Create Instructor and student user guides
- Proposal defense
- Pilot the system using volunteer test group of graduate students and professors
- Pilot the system in Kevin Matthews MIS 216 class Summer Session 1
- Record results
- Refine workflows
- Update documentation

- Formulate conclusion
- Final Defense

6.2 Scope

The scope of this project included configuring a single server install of TFS that acted as the central repository for students work in the ISOM department. The work item task workflows were customized for a generic class. User guides were created for both students and Instructors as pdfs (Appendix Q). There were four user guides for the students including connecting to TFS and mapping source control, working with work items, submitting work item task for help, and submitting work item task for grading. For Instructors there were user guides for how to setup a class, how to use work item templates, how to setup team queries, and how to setup and structure my queries.

6.2.1 What was included (specifics)

A virtual machine was configured with Windows Server 2003, SQL Server 2008 R2, IIS 6.0, SQL Reporting and Analysis Services, Team Foundation Server 2010, and SharePoint version 3.0. This acted as the central repository for all students' work in the ISOM department. A collection was created for classes, and for each class the instructor created a team project with the department, class number, and class section. A documented process was provided for the initial class setup. This included creating a folder for each student in the class named using their UNCW e-mail and assign permissions for each student. Workflows were defined for how students received assignments. This was done by creating a user story for the project and linking child work items for each student.

6.2.2 What was not included (specifics)

TFS comes with its own API which allows you to write scripts to automate tasks in TFS. This is particularly interesting as it could greatly reduce the amount of effort by the instructor to perform the initial class setup and when creating work items. TFS could have been integrated with Entropy which would have allowed the instructor to click a button to setup a class in TFS using their roster from Entropy. This would have run the custom script which could create all the necessary folders and permissions. This however, was outside the scope of this project. Also, the workflows and processes that were defined were aimed at the entry level classes and were meant to be generic across the classes. While custom tailoring of workflows may be needed to be defined for other classes, this was also outside the scope of the project. TFS also comes with Lab management which is a powerful tool for creating test environments and running cross platform testing. This was not included in the scope. Finally, unit tests could have been used to automate a substantial portion of the grading but it was outside the scope of this project.

- Customized work item tasks for every class
- Integrated with Entropy
- Unit Tests

6.2.3 Deliverables

- Single Server install of TFS10

- Documentation of installation
- Documentation of how to use the system
- Customized Work Item Tasks for a generic class
- Student user guides
    - Connecting to TFS and mapping source control to local drive
    - Working with work items
    - Submitting work item task for help
    - Submitting work item task for grading
- Instructor user guides
    - Initial class setup
    - How to use work item templates
    - How to setup Team Queries
    - How to setup and structure My Queries

6.3 Timeline

The proposed timeline for this project will include the proposal on the week of May 1, 2011, and the experiment took place summer session 1. The final defense took place on April 24, 2013.

6.3.1 Specific dates, milestones

Time	Activity	Activity Details
4 weeks	Design Server Architecture	• Determine software • Determine hardware
2 weeks	Set up accounts	• Determine accounts needed • Document and create accounts
2 weeks	Install and configure server	• Install Windows Server 2003 • Install IIS • Install SQL • Install TFS • Configure server
4 weeks	Analysis of As-IS	• Instructor Grading workflows • Student workflows
5 weeks	Analysis and design of As-Planned	• Instructor grading workflows • Student workflows
8 weeks	Design TFS structure	• Set up collections and file structure for department • Setup Team Project structure for classes

5 weeks	Implement TFS Summer Session 1	• Use TFS in undergrad setting • Use TFS in graduate setting • Use TFS to simulate process using instructors and fellow peers
3 weeks	Write Final Defense Paper	• Record results and conclude on the project • Create PowerPoint presentation

The actual execution of this project was not sequential as it appears in the timeline above. All of the activities up to the design TFS structure took place between approximately January 1, 2011 and June 30, 2011. Many of these activities were done in parallel as the system analysis drove the planned design. By far, the design of the implemented system took the most time due to the fact is was a trial and error approach. For each trial, planning, implementing, and testing had to occur which was very time consuming. A major milestone was in April of 2011 at the WITX conference. I had time to talk with Josh Wright who led me down the path of the final design. By the time the pilot started in summer session 1 in June 2011, all of the work was completed up to the pilot. Due to work obligations, no work was done on the final defense paper between August 2011 and January 2013. However, the original time estimates for the final defense paper were accurate once I was able to find time to work on it.

6.4 Resources

For this project I needed a server to host the virtual machine for the TFS server. I also needed the software and licenses to install TFS. I needed the help of my committee to guide me through this process and additional input from other faculty.

6.4.1 Who do you need?

Name	Responsibility
Dr. Douglas Kline	Capstone Chair
Dr. Tom Janicki	Committee member – process
Dr. Guinn Curry	Committee member
Josh Wright	Committee member – consultant, TFS expert
Kevin Matthews	Instructor for test pilot
Dr. Devon Simmonds	Input on other Software Engineering tools

6.4.2 What do you need?

Software	Functionality
Visual Studio 2010	Development tool to connect with TFS environment
Team Foundation Server 2010	Development Environment

SQL Server 2008	Database Engine
SQL Reporting Services	Database Reporting tool
SQL Analysis Services	Database Analysis tool
Windows Server 2003	Operating system for server
SharePoint version 3.0	Team collaboration tool
IIS 6.0	Role and authentication management

6.4.3 Critical resources?

My committee members and the help from faculty were essential to my success on this project. The software needed to setup TFS was critical because without it I could not proceed with this project. Lastly, the server that the TFS virtual machine was running on is very critical. Since there was a test pilot included in this project, if the host server crashed then I would have needed a backup server that the TFS virtual machine can be moved to. Any failure of the server in the middle of class would have resulted in my pilot failing.

6.5 Risks

There are several risks involved with this project. There was the risk of the host server crashing during the test pilot. There was also the risk that I would not be able to find an Instructor who was willing to do the test pilot in their class. With TFS there were risks that the system would not allow me to create a structure that would work in an educational setting.

6.5.1 Specific risks

- Host server crashing
- Failure to find Instructor willing to do the test pilot
- Failure to structure TFS for an educational setting

6.5.2 Mitigation methods

For the risk of the host server crashing, a backup of the TFS virtual machine needed to be made and stored on a separate machine and personal external hard drive. To mitigate the risk of not being able to find an Instructor for the test pilot I needed to talk with everybody involved in the ISOM department and get help from my committee if needed. If an Instructor could be found, then I could use my committee members and find graduate volunteers to do a pilot. This risk of being unable to structure TFS to work in an educational setting will be mitigated by receiving extremely valuable advice about the system and a large amount of research.

7 Predictions

7.1 Predication 1

The use of TFS will substantially reduce the amount of time spent on non-learning related items for both students and instructors.

7.2 Predication 2

With the use of work item tasks for assignments, students will receive more feedback. Through the experiment, reports can show how many times the work item tasks went back and forth between the students and Instructor when students needed help. They can also show the Details area of the work item task which documents the students' questions and the instructor's response.

7.3 Predication 3

Students will be better prepared to utilize the professional programming environments required by companies. At the end of the experiment the students will take a survey with questions about using the environment. With their responses I will be able to determine if they were able to easily work in the environment and if they understood some of the basic concepts. If students can successfully use this environment and understand the basic concepts, then they have been exposed to a professional programming environment and can work in it.

8 Experiment

8.1 Description

For a test pilot using TFS in the classroom, Professor Mr. Matthews agreed to try it in his Summer Session 1 MIS 213 class. The class size was roughly twenty students which consisted of novice programmers. For many of them, this was their first introduction to programming. One of the requirements of using TFS in the classroom is that students need to be able to understand how to connect and access TFS. On the first day of class after Mr. Matthews introduced the class, students were quickly introduced to VS10 (Visual Studio 2010). By following the professor walk-through the steps on the overhead projector, all of the students were able to setup the connection to TFS and connect to the class's team project.

There was a brief discussion on source control and how it would be used in the class. The students were then shown how to get to their workspace in source control. From there, students were shown how to map their workspace in source control to a local hard drive or usb drive. There were little to no problems getting the entire class setup to start on the first assignment.

The next step was to show the students how to access work items within TFS. All of the assignments are assigned as work item tasks and students were shown how to use the team queries to view their assignments. The very first assignment was small but walked through all of the steps necessary to create a new project, check-in to source control, change the status of their work item task, and perform a final check in for grading. By the end of the first day, all the students in the class were able to successful complete their first assignment. Kevin and I sat down and went over how to access and grade the students' assignments with TFS. Kevin needed very little instruction and was able to quickly access and grade all the students' submissions.

The next class the students were reminded how to connect to TFS, map their source control, and access the team queries. By accessing the team queries, the students could see their grades as soon as Kevin completed the work item task for an assignment in TFS. From this point forward, students were able to connect to TFS on their own, get to their assignments, request help, submit their assignments, and view

their grades and feedback. I had initially planned on sitting in for the entire class to help mitigate the risk of students not being able to understand the tasks required. However, after the third class I was no longer needed as the students were able to do everything required without additional assistance.

8.2 Results

The test pilot was very successful. The students were able to understand what they needed to do in order to use PeTE. The time it took Mr. Matthews to grade the assignments was reduced by 50 percent from the time it was taking him to grade before PeTE. In addition, once the class started the TFS server required no additional configuration, modifications, or maintenance of any kind. However, one of the greatest benefits of this experiment was the exposure of TFS, source control, and work item tasks to students. These are skills that are highly sought after by employers that most students do not acquire until after they are in the workplace.

At the end of the class, a survey was given to the students (Appendix N). The survey consisted of 3 sections based on a likert scale and an open ended comments section. Refer to Appendix O for the spreadsheet of the student results and Appendix S for the graphs of the student survey results. The three main sections asked questions relating to the core concepts used based on level of difficulty, level of understanding, and how useful the students found some of these core concepts. The results from the survey illustrated that the students did not feel overwhelmed and overall found PeTE to be useful (Appendix T). None of the students found the material to be extremely difficult. All responses ranged from average to very low in perceived difficulty. For the section that evaluated the students' level of understanding on the core concepts, only one student had a below average understanding of source control. All other students ranged from average to very well. Source control was the area that students struggled in the most. The areas that students felt they understood the best was working with the work item tasks and checking in and out of source control.

The section that asked students how useful they found the concepts being used, the responses ranged from very low to very high. However, only 4 students found any of the concepts to have no use. The user guides were one of the items that ranged from no use to very useful. In a discussion with Mr. Matthews, he speculated that he thought the reason that the students found them to be of no use was due to the fact that they did not need them. Source control was the other area that students found to be less useful. In the interview with Mr. Matthews, he noted that while he discussed source control at the beginning of class; due to the simplistic nature of their first applications, many students were not able to experience the benefits of source control first hand.

While most students did not leave any additional comments in the open ended section, there were a couple of comments. The comments were positive and the generally commented on how nice it was to have a central repository for all of their work where they could get to from anywhere. It is important to note that the students had very little problem connecting to in class, students did experience some difficulty connecting from home. This was not due to the TFS server, but the fact that students had to create a vpn connection in order to be able to connect.

8.3 As-planned versus as-executed

I spent so much time preparing for this project that once the class started, everything went as planned. None of the configuration or the workflows needed to be modified. The one thing that I had not planned on was the server that was running the virtual server for this project to fail the night before the first day of class. It was about two hours before the first day of class was supposed to start that I discovered that the memory had gone bad on the host server and my virtual TFS server was inaccessible. Luckily with the help of Dr. Kline, we were able to get the virtual server moved to another host machine and get it booted up only a few minutes before class started.

I did create the workflow for the work item tasks used for student's assignments to allow for the work item to go back and forth between the instructor and the student. If the student needed help outside of class they could change the status of the work item to assign it to the instructor indicating that they needed help. To the nature of summer classes meeting every day and large portion of student work being done in class, this was not needed.

9 Predictions

9.1 Prediction 1

"The use of TFS will substantially reduce the amount of time spent on non-learning related items for both students and instructors."

This was more observable for the instructor. With all of the students' code being in a central repository, there was no need to retrieve the students file. The student merely gets current on the team project and has access to all of the students' workspaces in TFS. This was a huge time saver as prior, the instructor would have had to spent time retrieving all of the students' files as well as setting them up so they could be graded.

For students, they had a more controlled workspace with an easier way to submit their work. This saved time for the students on any time lost dealing with multiple versions of projects. It also made it easier for students if they needed help outside of class. They could check in their code, and then meet with the instructor during office hours. The instructor could quickly and easily pull up the students' project on their machine so they could review it with the student. However, with this being the first introduction to programming for most of the students, there was no baseline to compare with.

9.2 Prediction 2

"With the use of work item tasks for assignments, students will receive more feedback."

Because this was a summer class, students tended to get feedback directly in lab from the instructor. So this prediction did not come to pass.

Through the experiment, reports can show how many times the work item tasks went back and forth between the students and Instructor when students needed help. They can also show the Details area of the work item task which documents the students' questions and the Instructor's response.

The experiment was executed on Kevin Matthew's summer session 1 class. Summer classes meet four days a week for two hours a day. This resulted in much of the student's work being done in class or immediately after class. While students could submit a work item task to receive help from the instructor, help was typically sought during or after class. This resulted in the work items not going back and forth between the instructor and the students.

While the method for submitting a work item task to receive help from the instructor was not used, other benefits were realized. Being able to quickly access the student's projects, more time was available to review the student's assignment and provide feedback. The instructor also had the ability to add comments directly in the student's code and check this into the source. Students were then able to get current on their workspace and view these comments injected directly into the code.

9.3 Prediction 3

"Students will be better prepared to utilize the professional programming environments required by companies."

Through this pilot, students used the following on a daily basis for five weeks:

- Team Foundation Server
- Source Control
- Work Items
- Team Queries

These are all professional level skills that were otherwise not taught or experienced in the normal course format.

At the end of the experiment the students will take a survey with questions about using the environment. With their responses I will be able to determine if they were able to easily work in the environment and if they understood some of the basic concepts. If students can successfully use this environment and understand the basic concepts, then they have been exposed to a professional programming environment and can work in it.

The students were able to understand how to use Visual Studio to connect to the TFS server and retrieve their assignments as work item tasks. In a professional development environment, these are the tools and similar workflows that they use. At the end of the class a survey was given with basic questions on the concepts that were being used in this project (Refer to Appendix N). There are three sections in the survey, the first section asking students to rate on a scale of difficulty connecting to the TFS was, mapping source control, and other related concepts. The second section asks student to rate their level of understanding for these concepts. The last section asks student to rate on a scale of their level of understanding of these concepts.

On average students rated the difficulty as low, their level of understanding was well, and on average they considered these concepts to be useful. Refer to Appendix O for the results of the student survey. In the section where the students rated the level of difficulty, all students answered average or better.

The area that students found to be the most challenging was source control. In the section where students measured their level of understanding, all except one response was average or better. Once again, students viewed source control as a challenging area. In the last section using the likert scale, students were asked to rate how useful they found the tools used in this experiment. These ranged from no use to very useful. The areas that were viewed as the least useful were version control and the user guides. In a discussion with Mr. Matthews, he speculated that the reason the user guides were regarded as having little use was due to the fact that the students did need them. For source control, he discussed that it wasn't until the last project that the students were able to first hand see the benefits of the project. This was due to the simple nature of the assignments, with the last assignment being more challenging.

10 Commentary

When I first started this project, TFS appeared to be huge and complex. I was also using TFS in a way that it was not designed for. At the time of this project, no one was attempting to use TFS in this way so there was no documentation that I could use to aid in my approach. This increased the overall difficulty of the project. To reach the final design of the system it took many approaches. Each approach required planning, implantation, and testing which was very time consuming. It was only through repeated trial and error that the final implemented configuration was reached.

The more I worked with it, the more manageable that TFS became. To setup TFS there are a lot of components that have to work together, but using the online documentation it is an achievable task. Once TFS is setup, it is easy to connect to and use with VS (Visual Studio). From there, it is understanding how TFS works out of the box, and what you can do to tailor it to your needs. Through trial and error as well as advice from people who have worked with TFS, I learned that you want to use TFS out of the box as much as you can. The more you customize and step away from the standard, the more complex you can make things for yourself, and the people that use TFS.

When this project started, there was a general concern that students in the lower level classes would be unable to use the system. From the test pilot I learned that the students were very quick to understand how to get into Visual Studio 2010 and connect to TFS. They were also quick to understand how to map their source control to the local file system. Once they did all of this, there were questions as to why and how to use source control. I think that there still needs to be additional instruction on source control and best practices for check ins.

Mr. Matthews allowed me to test this PeTE in his Summer Session 1 MIS 216 class. He did go on to continue using PeTE for his MIS 216 classes for fall semester 2011, Spring and Fall 2012, and his last class teaching at UNCW Spring 2013. During discussions and a formal interview with Mr. Matthews (Appendix R), he expressed interest in using PeTE to aid in teaching at other institutions. He had all of the knowledge needed to set it up to use for class, but wasn't comfortable with being able to install everything required. A hosted TFS solution would allow him to avoid the installation while being able to configure it for the classroom at other institutions.

11 Future work

In order for an instructor to use the system, there is additional setup and configuration needed at the beginning of a semester to setup their class. However, this is negligible when you observe how much time it can save an instructor throughout a semester. One of the risks of this project was faculty buy in. To ease the process for an instructor to adopt this system and use it in their classes, scripts could be developed to automate the class setup. This would make it easier for instructors and remove the burden of the initial setup.

One of the areas that I wanted to address in this project was peer programming. As a programmer, we are often required to work with other developer's code. However, the students typically work alone and never review other student's code. Using TFS, we have the ability to introduce such a concept into the classroom. A shared workspace could be set up in the class's source control for two or more students. Having assignments where students had to work together on a project would have many benefits. Students would learn how to work collaboratively on a project with another programmer. They would be forced to deal with check-ins on the same files and having to resolve conflicts and merges on check-ins. Students would also be exposed to having to read and interact with other programmers' code.

In addition to getting students to evaluate code that was not written by them, bugs was an area that would add benefit to the class. As a developer, a lot of time is spent finding and fixing bugs. Using PeTE, the instructor could create a program with a list of known bugs and expose it in source control to all students. Work item bugs could be created for each known bug, and the first student to find and fix the bug could attach the work item bug to their check in and receive extra credit. This would make it competitive for the students and give them a chance to review somebody else's code.

Another area where automation could reduce the amount of work needed by instructors would be unit tests. If time was spent before hand, unit tests could be written for each assignment. This would change the grading process drastically. When the instructor is ready to grade, they could build the students project applying the unit tests for the assignments. The build output would be viewable by the student as well as the unit tests that passed and failed. This would allow for students to go back and fix the unit tests that failed, submitting the assignment as many times as they wanted. This process would also ensure that grading is impartial, consistent, and greatly reduce the amount of time needed.

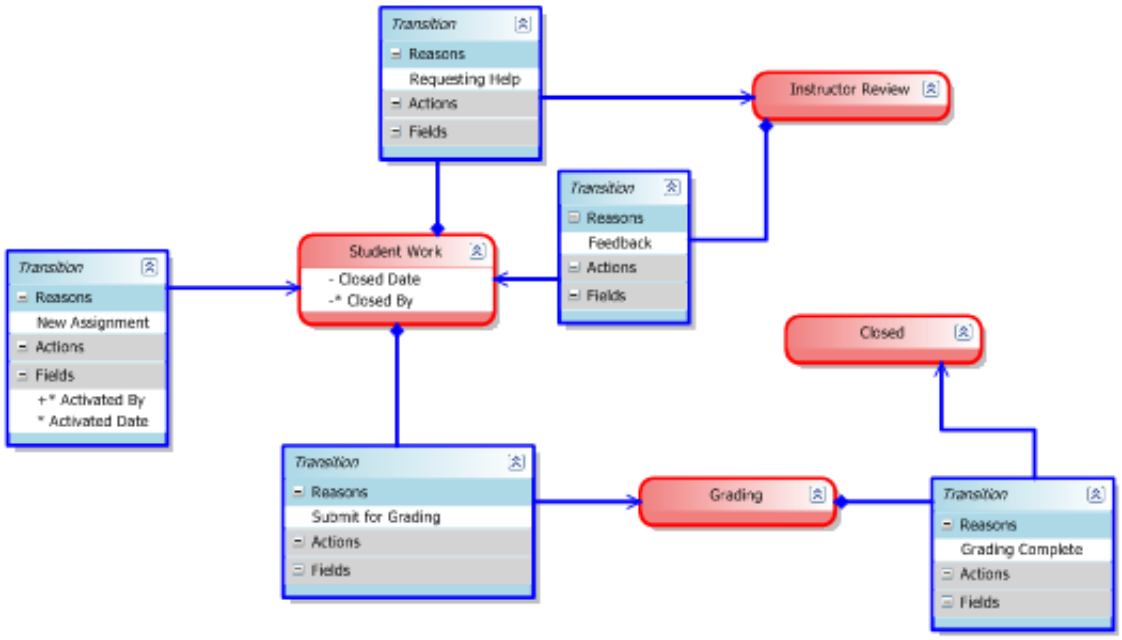
While Mr. Matthews used PeTE for all of his MIS 216 classes, spring semester 2013 is his last semester teaching at UNCW. Nothing needs to be changed on the server, but additional information is needed for the next instructor in the case that the new instructor wishes to use PeTE. The documentation needed for an instructor to setup the class can be referred to in Appendix P. There are also user guides for the students which can be referred to in Appendix Q. These user guides demonstrate how to connect to the TFS server, retrieving a previous version of a project, mapping source control, submitting work items for grading, and working with work items.

The workflows and processes that were used were created as generic as possible. However, the workflows and processes that worked so well for MIS 216 will not necessarily work for MIS 316 or MIS 413. Additional work would need to be done to evaluate the needs of these classes. This analysis would then in turn be used to construct new workflows if they were needed.

Work Cited

1. Microsoft. Visual Studio 2010. Team Foundation Server 2010. Overview. 2011 Archive. <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/team-foundation-server/overview>.
2. Microsoft. "Microsoft Visual Studio Team Foundation Server 2010 Datasheet(pdf)." http://Microsoft. com.
3. Coding Horror, programming and human factors. Jeff Atwood. August 16, 2010. <http://www.codinghorror.com/blog/2006/08/source-control-anything-but-sourcesafe.html>.
4. Distributed Version Control is here to stay, baby. Joel Spolsky. March 17, 2010. <http://www.joelonsoftware.com/items/2010/03/17.html>.
5. Microsoft Visual SourceSafe. May 7, 2011. <http://en.wikipedia.org/wiki/Source_Safe>.
6. Better SCM Initiative: Comparison, Version Control System Comparison. Shlomi Fish. January 19, 2011. http://better-scm.berlios.de/comparison/comparison.html#atomic_commits
7. Eran Ruso
8. (Illeris, 2000; Ormorod, 1995), Wikipedia, http://en.wikipedia.org/wiki/Learning_theory_%28education%29
9. The Conditions of Learning (4th Edition). Robert M. Gagne. 1985. New York: Holt, Rinehart and Winston.
10. Msdn. Work Items. http://msdn.microsoft.com/en-us/library/ms243845%28v=VS.80%29.aspx
11. On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process. Journal of Information Technology Education. Pdf. http://jite.org/documents/Vol5/v5p271-283Al-Imamy115.pdf.
12. Organizing Your Server with Team Project Collections http://msdn.microsoft.com/en-us/library/dd236915.aspx
13. Visual SourceSafe: Microsoft's Source Destruction System, Alan D. Smet, http://www.highprogrammer.com/alan/windev/sourcesafe.html
14. Software Configuration Management Strategies and IBM Rational ClearCase: A Practical Introduction, 2nd Edition. David E. Bellagio, Tom J. Milligan. May 23, 2005.

Appendix A – Proposed Student Work Item Task Workflow

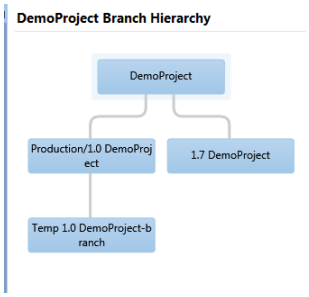


Appendix B – Features of Source Control Software

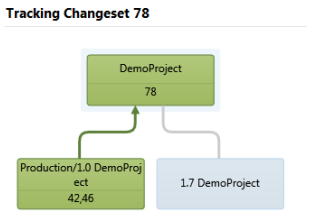
Features [edit]

Software	Atomic commits	File renames	Merge file renames	Symbolic links	Pre/post-event hooks	Signed revisions	Merge tracking	End of line conversions	Tags	International Support	Unicode filename support
AccuRev	Yes	Yes	Yes	Yes	Yes	Yes[citation needed]	Yes	Yes	N/A	Yes	Unknown
Bazaar	Yes	Yes	Yes	Yes	Yes	Partial[47]	Yes	Yes[28]	Yes	Yes	Yes
BitKeeper	Yes	Yes	Unknown	Unknown	Unknown	Unknown	Yes	Unknown	Yes	Unknown	Unknown
CA Software Change Manager	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ClearCase	Partial[29]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes[30]	Unknown
Code Co-op	Yes	Yes	Yes	No	Partial	No	No	No	Yes	Unknown	Unknown
Codeville	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
CVS	No	No	No	No	Partial	No	No	Yes	Yes	Unknown	No
CVSNT	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
darcs	Yes	Yes	Yes	No	Yes	Yes	N/A[31]	No	Yes	No	Unknown
Dimensions	No	Yes	No	No	No	No	No	No	Yes	No	No
Fossil	Yes	Yes	Unknown	No	No	Yes	Yes	Yes	Yes	Yes	Unknown
Git	Yes	Partial[32]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Partial[33]	Partial[34]
GNU arch	Yes	Yes	Unknown	Yes	Yes	Yes	Unknown	Unknown	Yes	Unknown	Unknown
IC Manage	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Unknown
LibreSource Synchronizer	Yes	Yes	Yes	No	Partial[35]	No	Yes[36]	No	Yes	Unknown	Unknown
Mercurial	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes[37]	Partial[38]
MKS Integrity	Yes	Yes	Yes	No	Yes	Yes[39]	Yes[40]	Yes	Yes	Yes	Yes
Monotone	Yes	Yes	Yes	No[41]	Yes	Yes, mandatory	Yes	Yes	Yes	Unknown	Yes
Perforce	Yes	Yes[42]	No	Yes	Yes	Yes	Yes[43]	Yes	Yes	Yes[44]	Yes[45]
Rational Team Concert	Yes	Yes	Yes	Yes	Yes[46]	Yes	Yes	Yes	Yes	Yes	Yes
StarTeam	Yes[47]	Yes	Unknown	No	No	No	Yes	Yes	Yes	Yes	Unknown
Subversion	Yes	Yes[48]	No	Yes	Yes	No	Yes[49],	Yes	Partial[50]	Yes	Yes
SVK	Yes	Yes	Yes	Yes	Yes[51]	Yes[52]	Yes	Yes	Yes	Yes	Unknown
Synergy	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes[53]
Team Foundation Server	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Vault	Yes	Yes	Yes	No	Yes	No	No	Yes	Yes	Unknown	Unknown
Visual SourceSafe	No	No[54]	Unknown	No	Yes	No	No	Unknown	Yes	Yes	Unknown
Software	Atomic commits	File renames	Merge file renames	Symbolic links	Pre/post-event hooks	Signed revisions	Merge tracking	End of line conversions	Tags	International Support	Unicode filename support

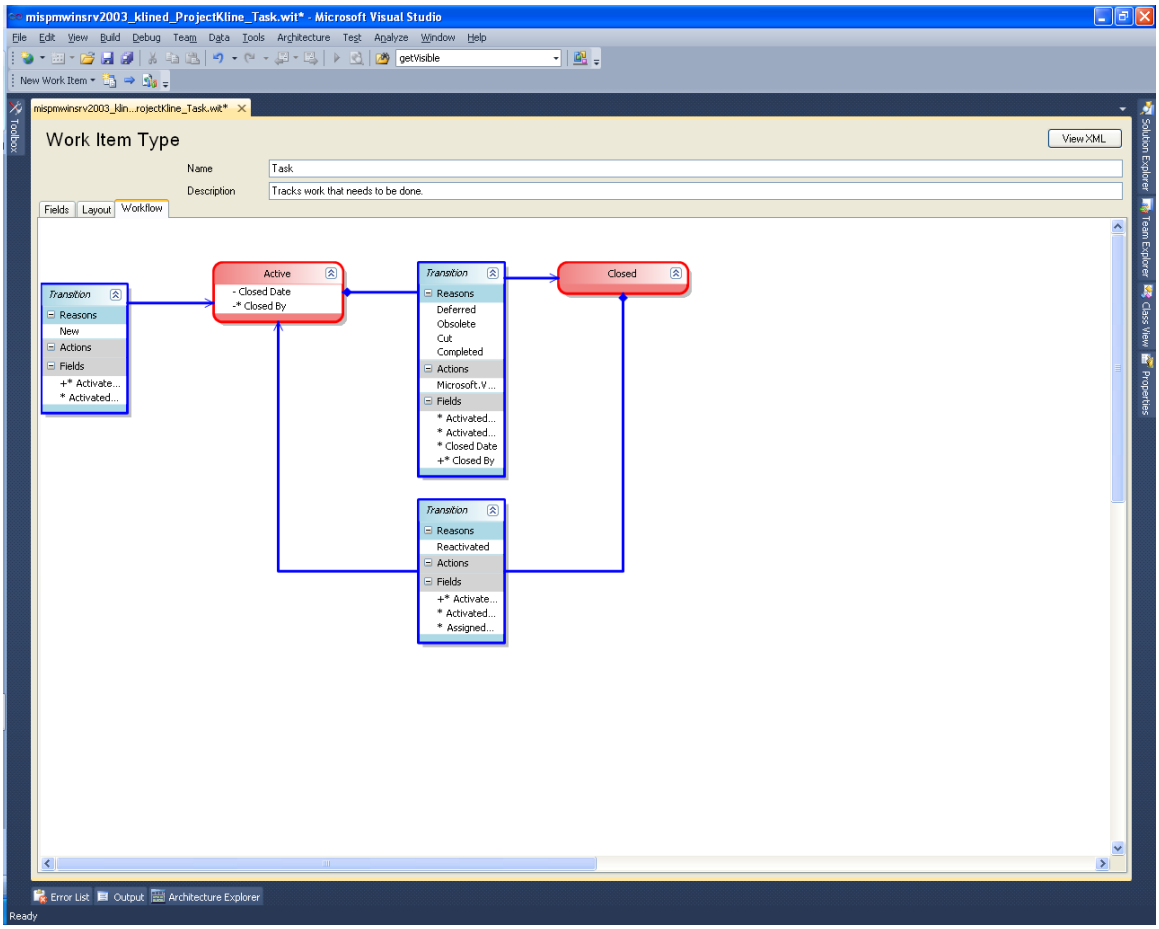
Appendix C – Branch Hierarchy



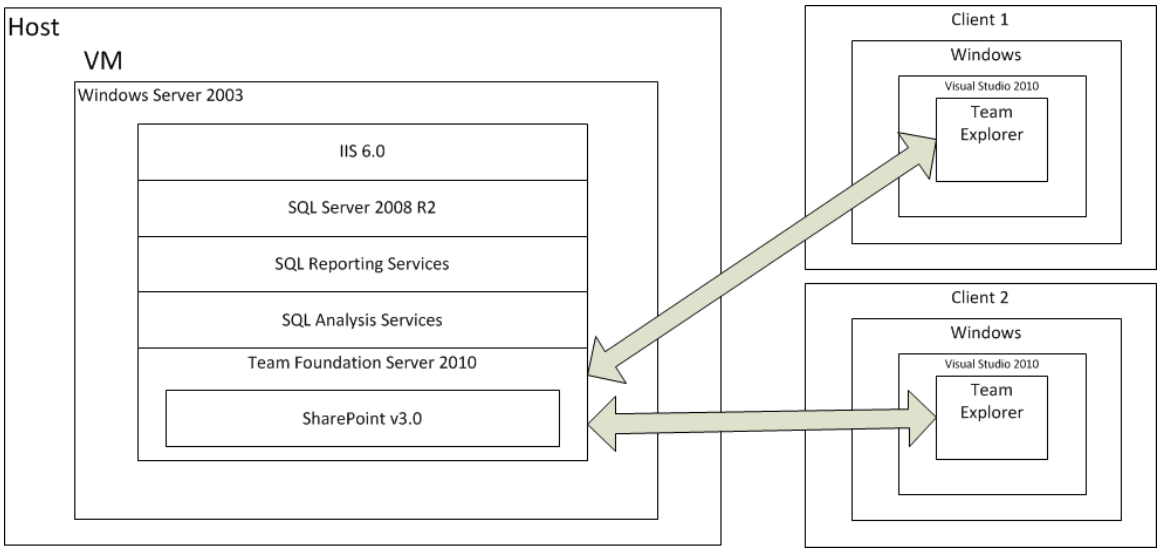
Appendix D – Branch Hierarchy Tracking Changesets



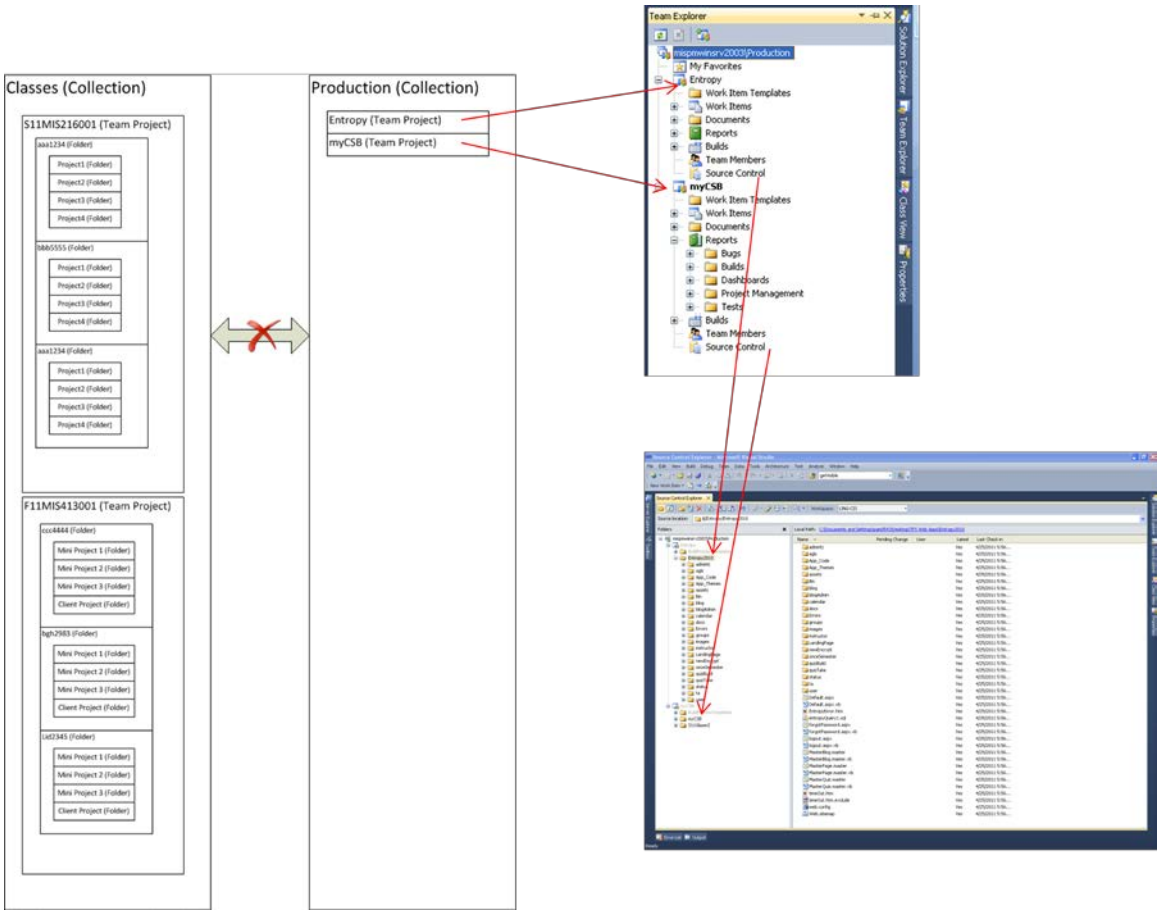
Appendix E – Basic Work Item Task Workflow



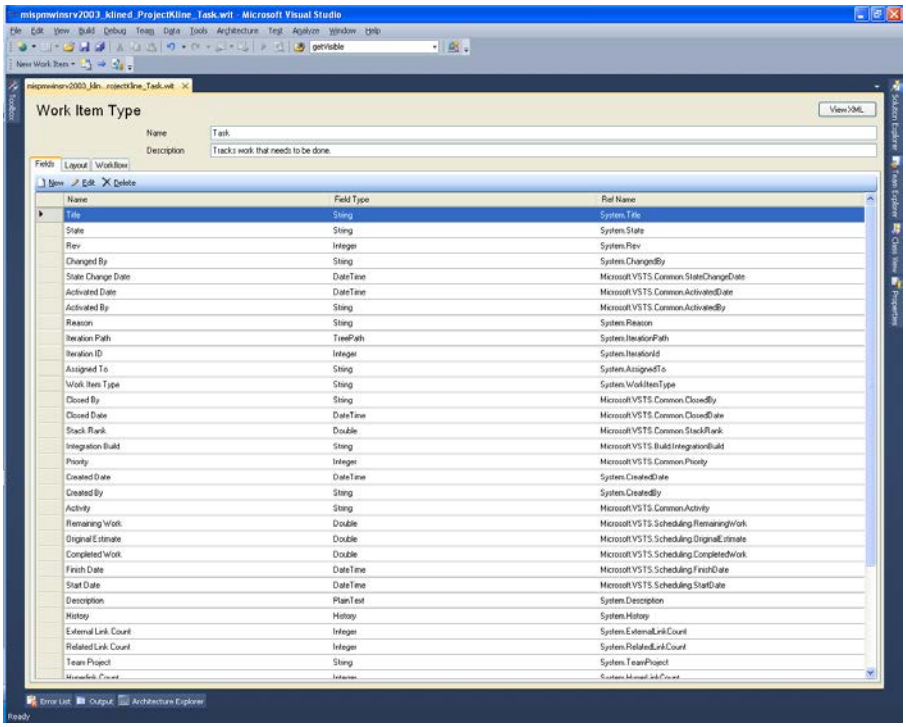
Appendix F – Server Architecture

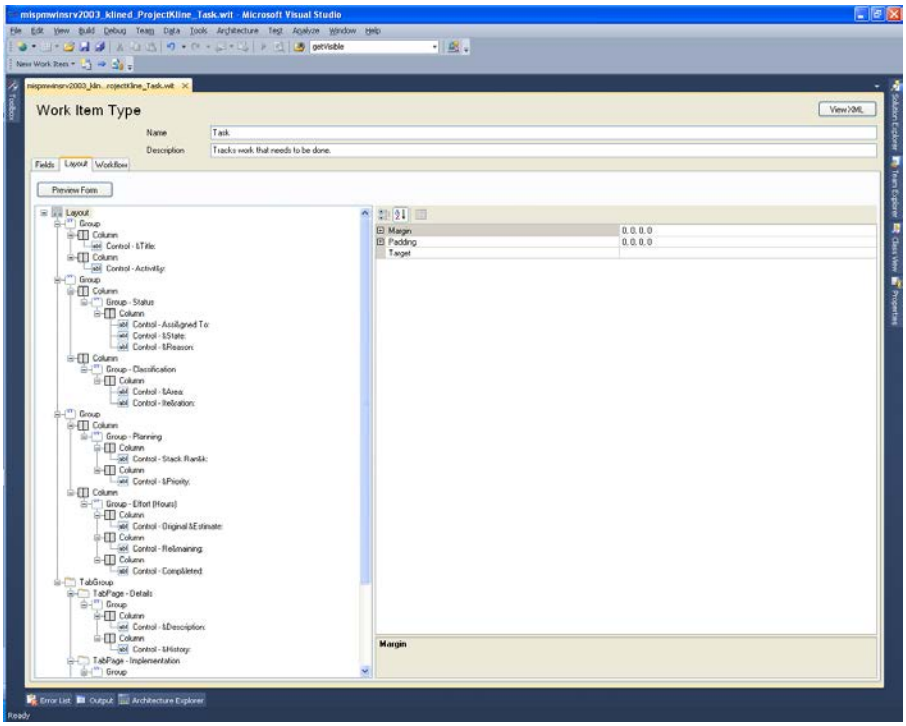


Appendix G – Team Foundation Server Configuration

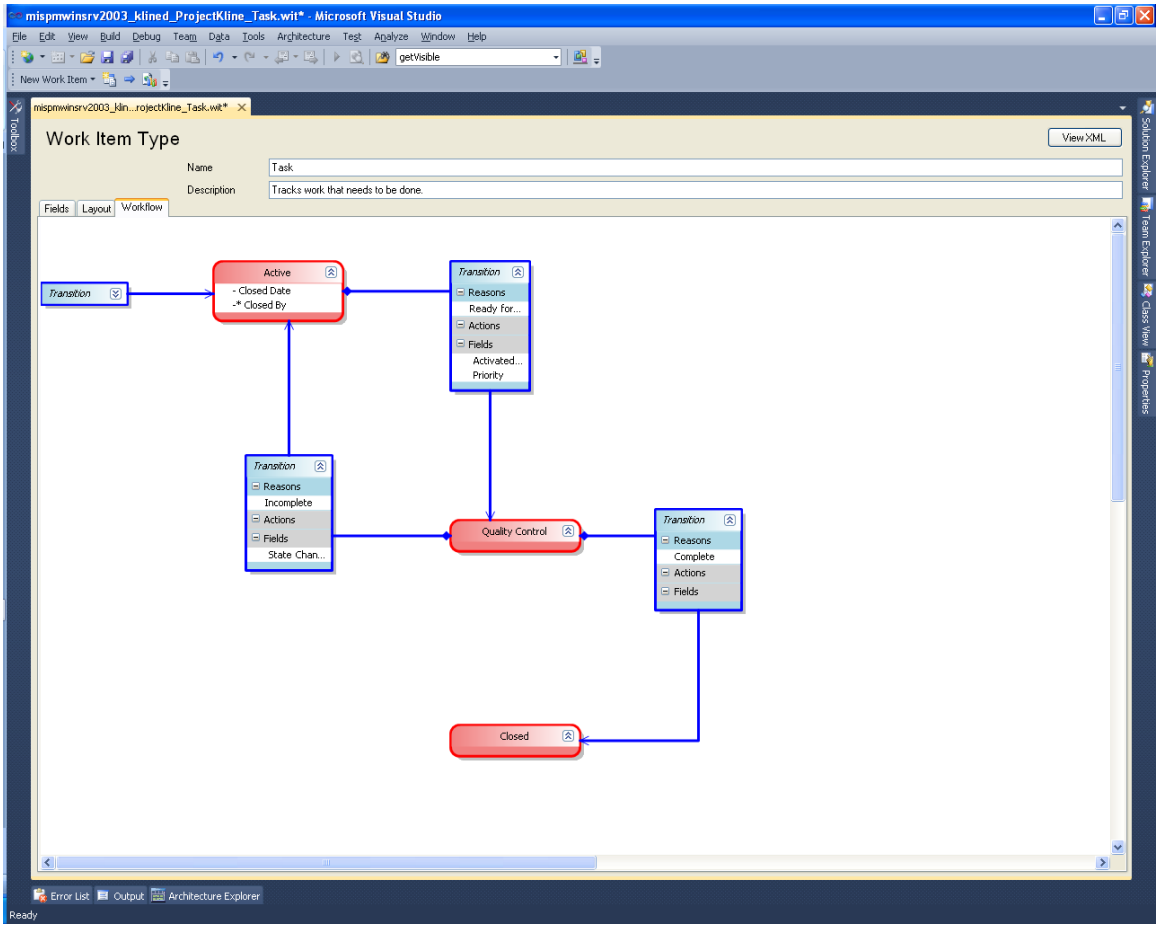


Appendix H – Power Tools for Customizing Workflows

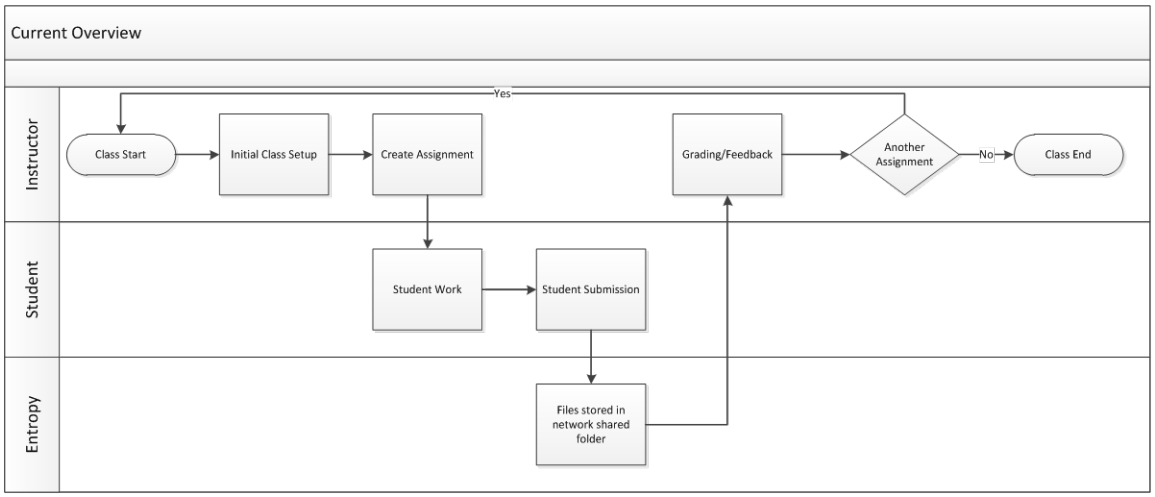




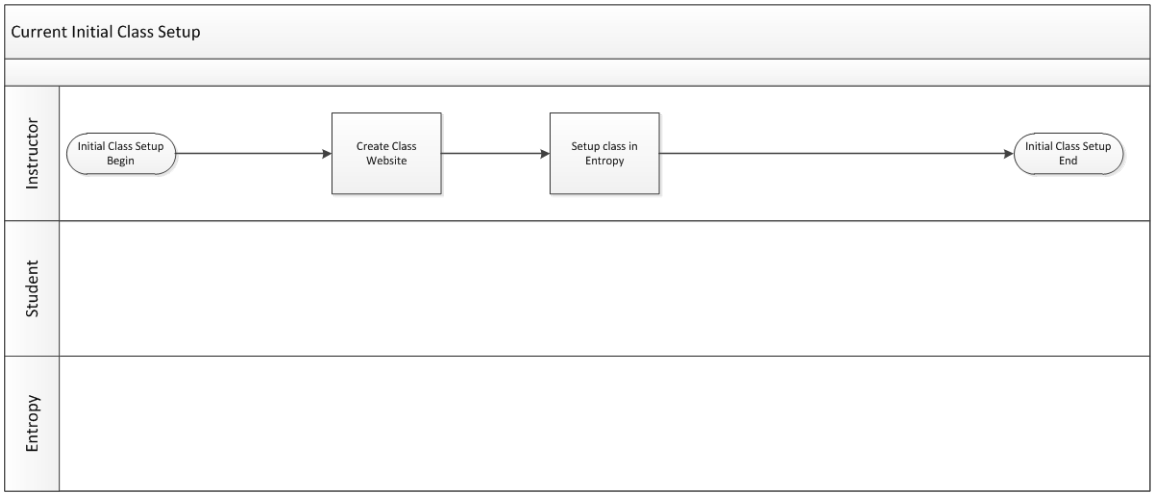
Appendix I – Power Tools for Customizing Workflows



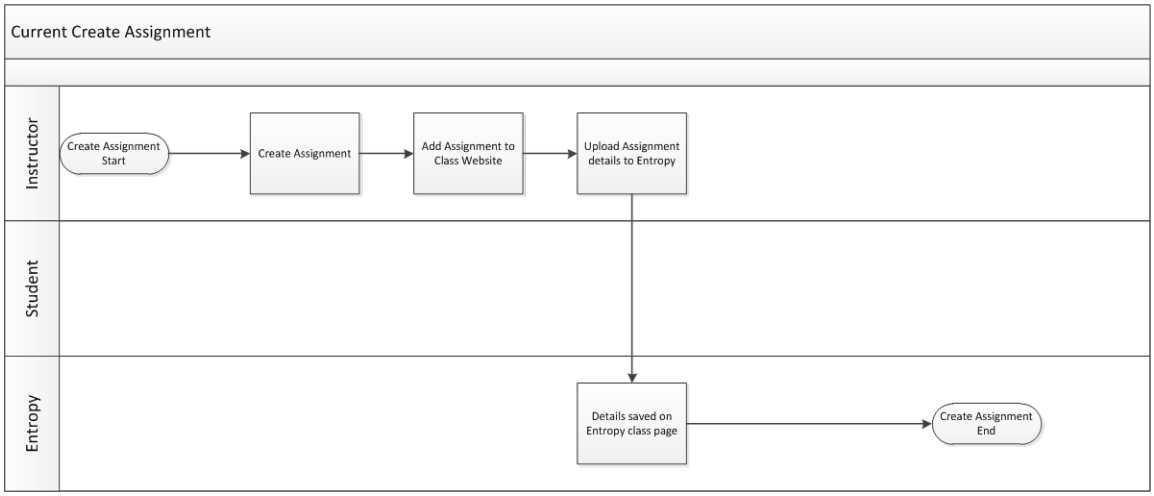
Appendix J.1 – As Is Overview



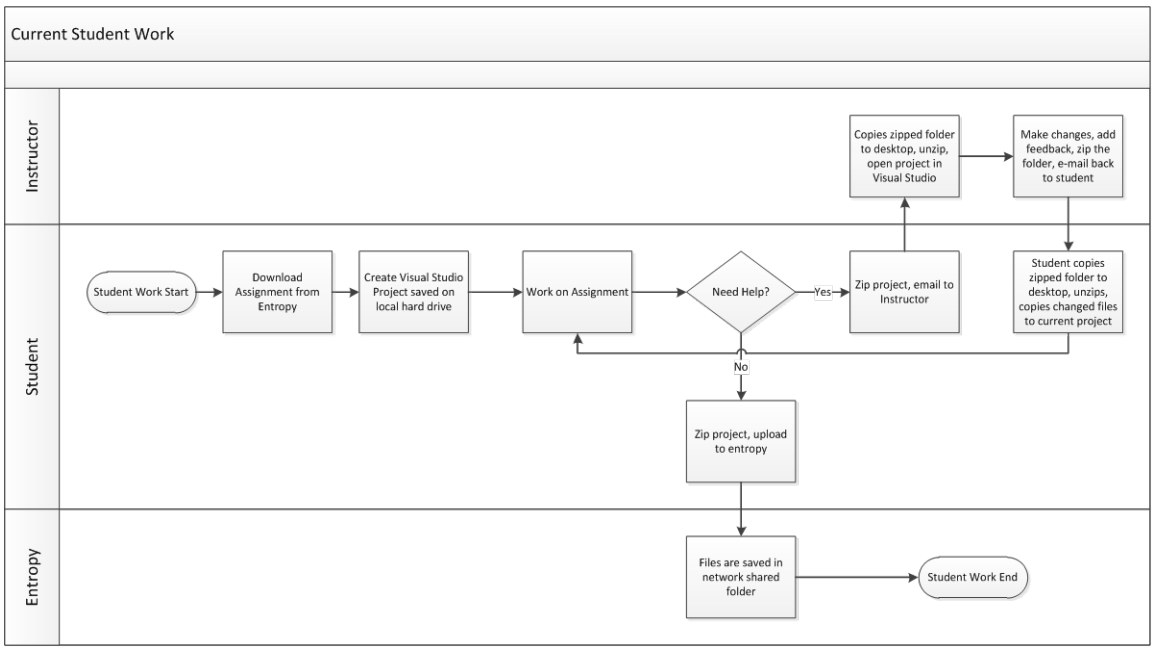
Appendix J.2 – As Is Initial Class Setup



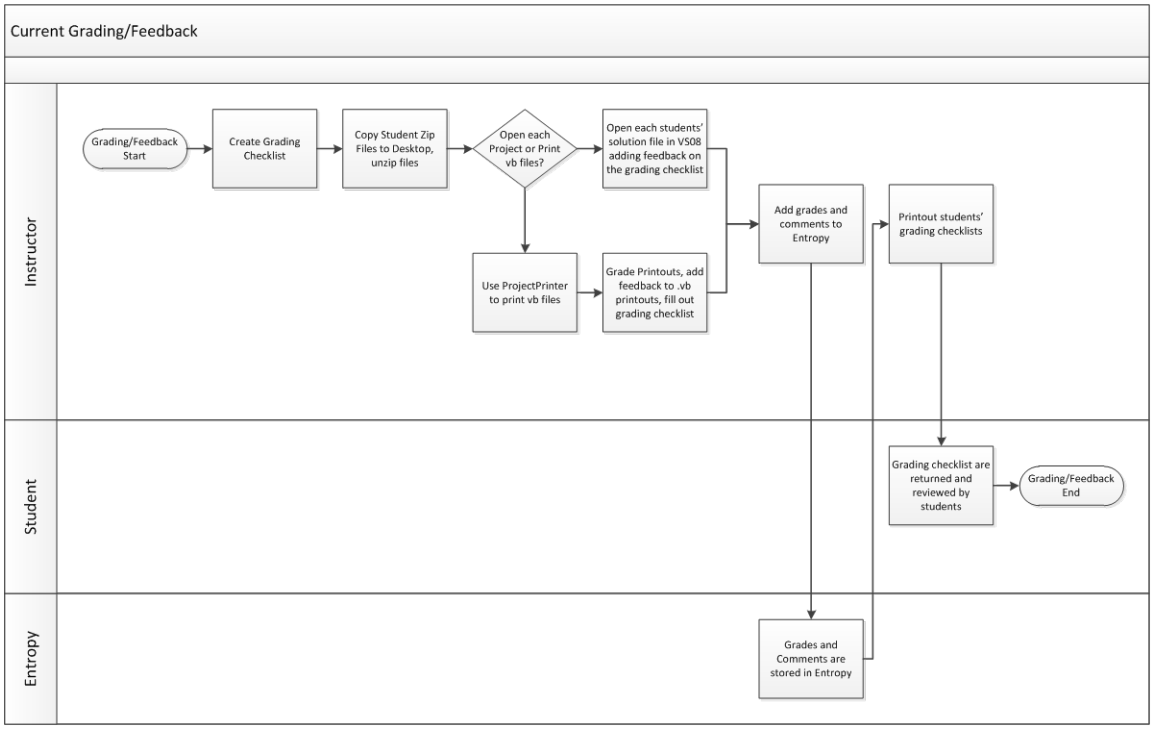
Appendix J.3 – As Is Create Assignment



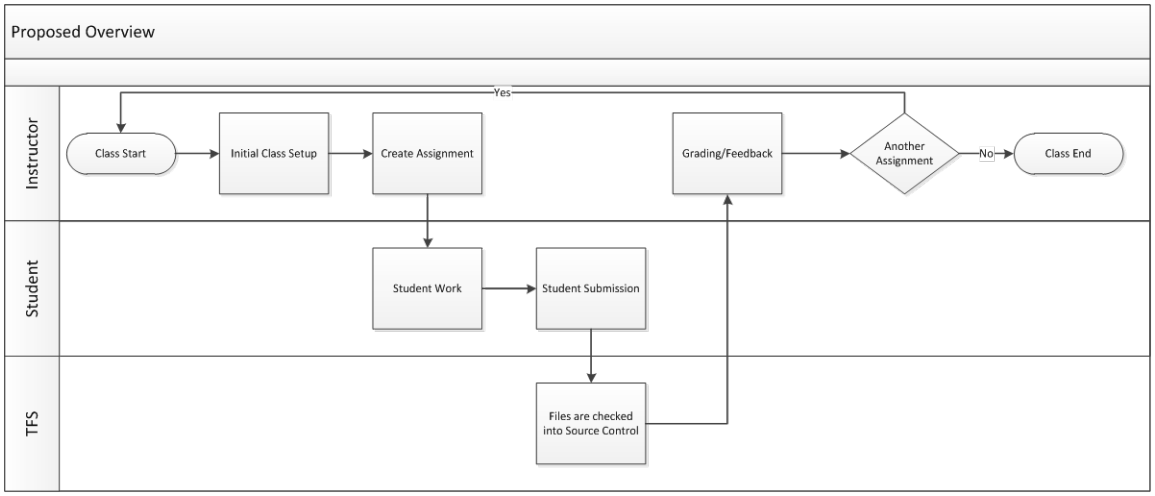
Appendix J.4 – As Is Student Work



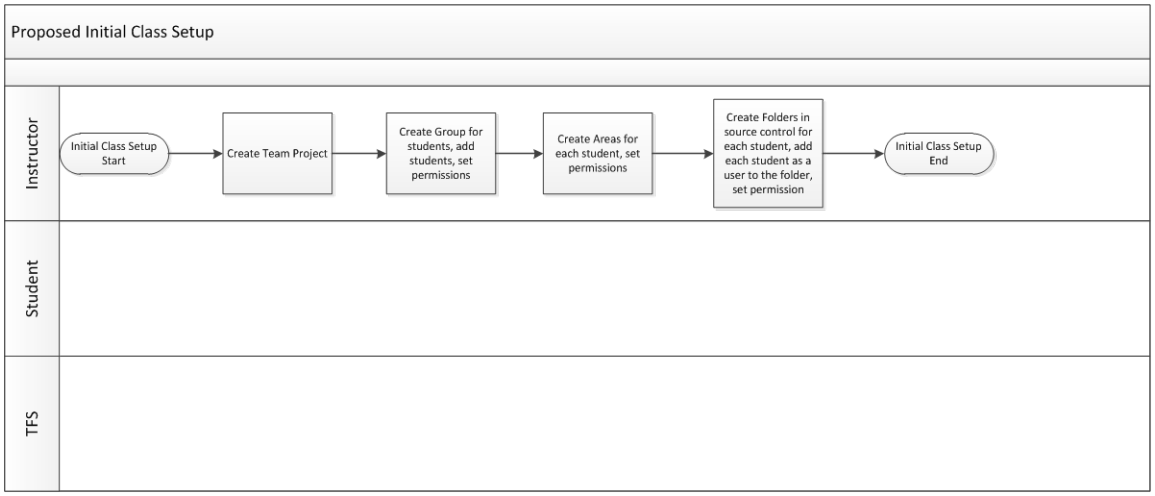
Appendix J.5 – As Is Grading/Feedback



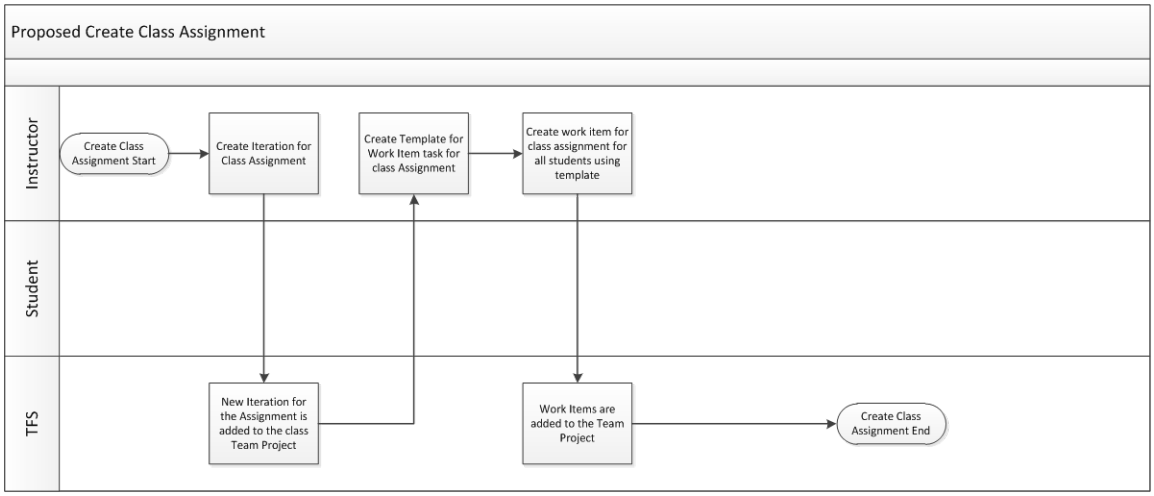
Appendix K.1 – As Planned Overview



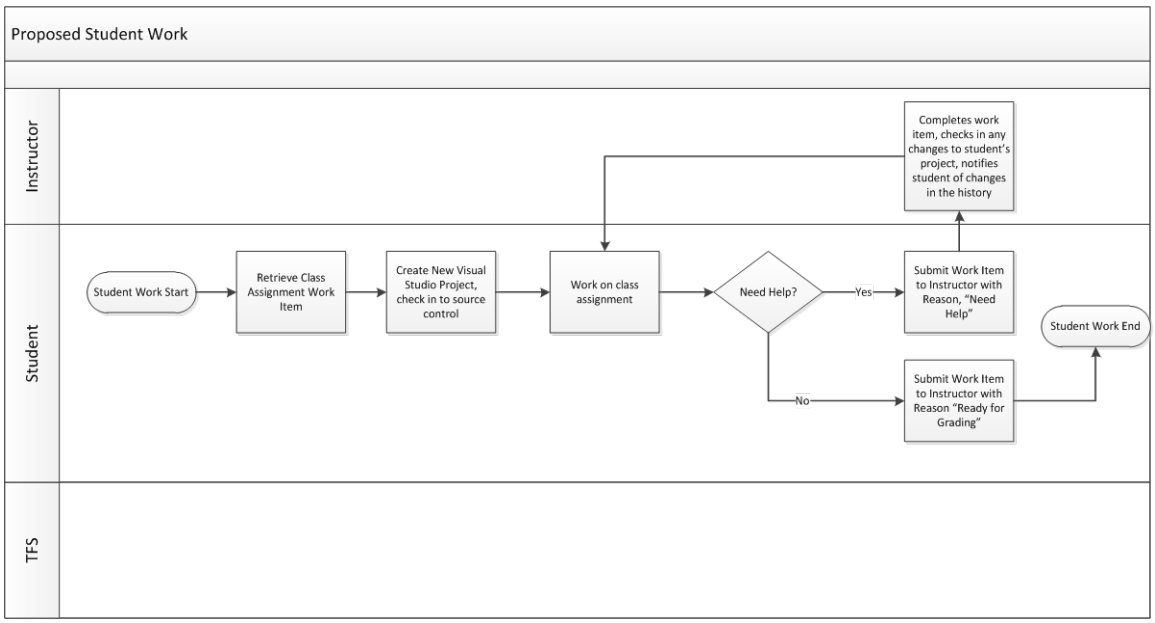
Appendix K.2 – As Planned Initial Class Setup



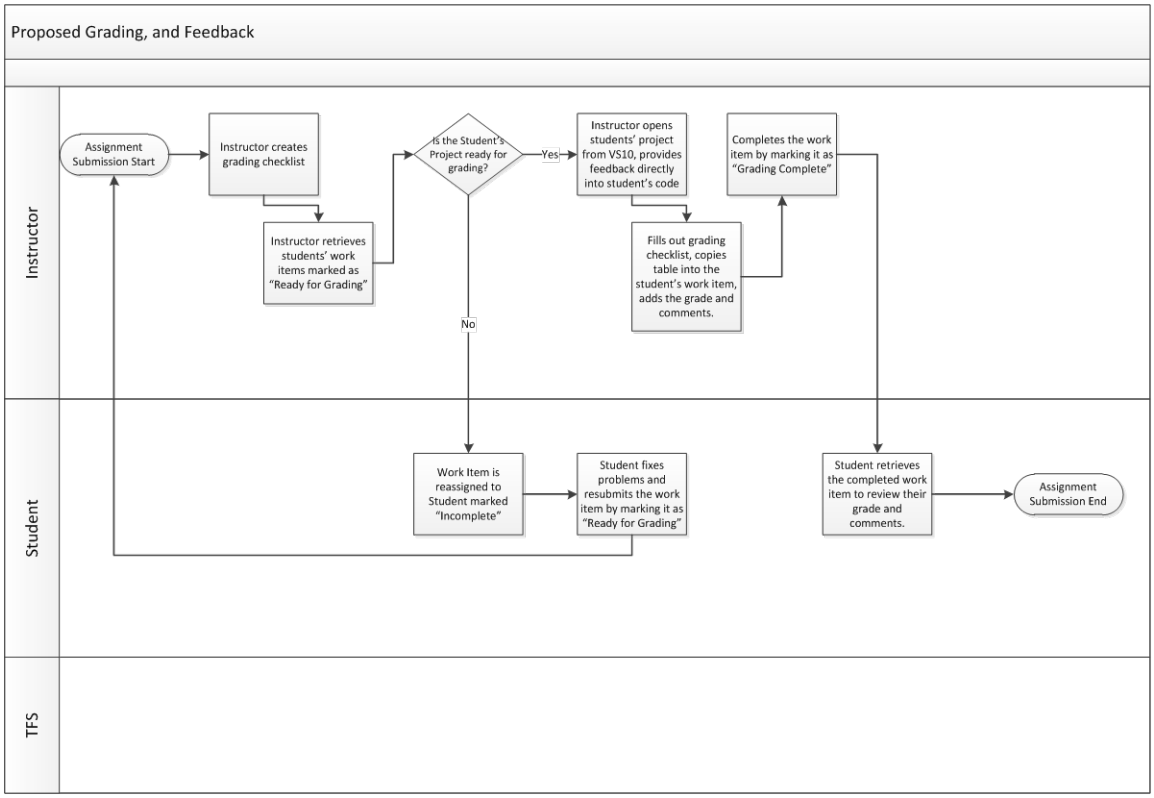
Appendix K.3 – As Planned Create Assignment



Appendix K.4 – As Planned Student Work

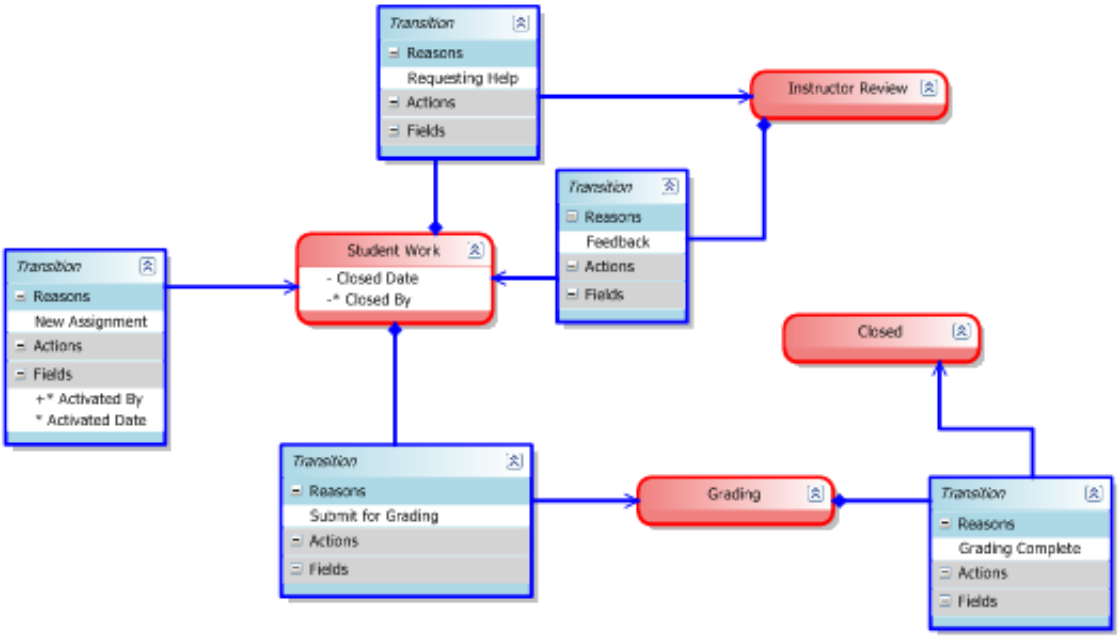


Appendix K.5 – As Planned Grading and Feedback

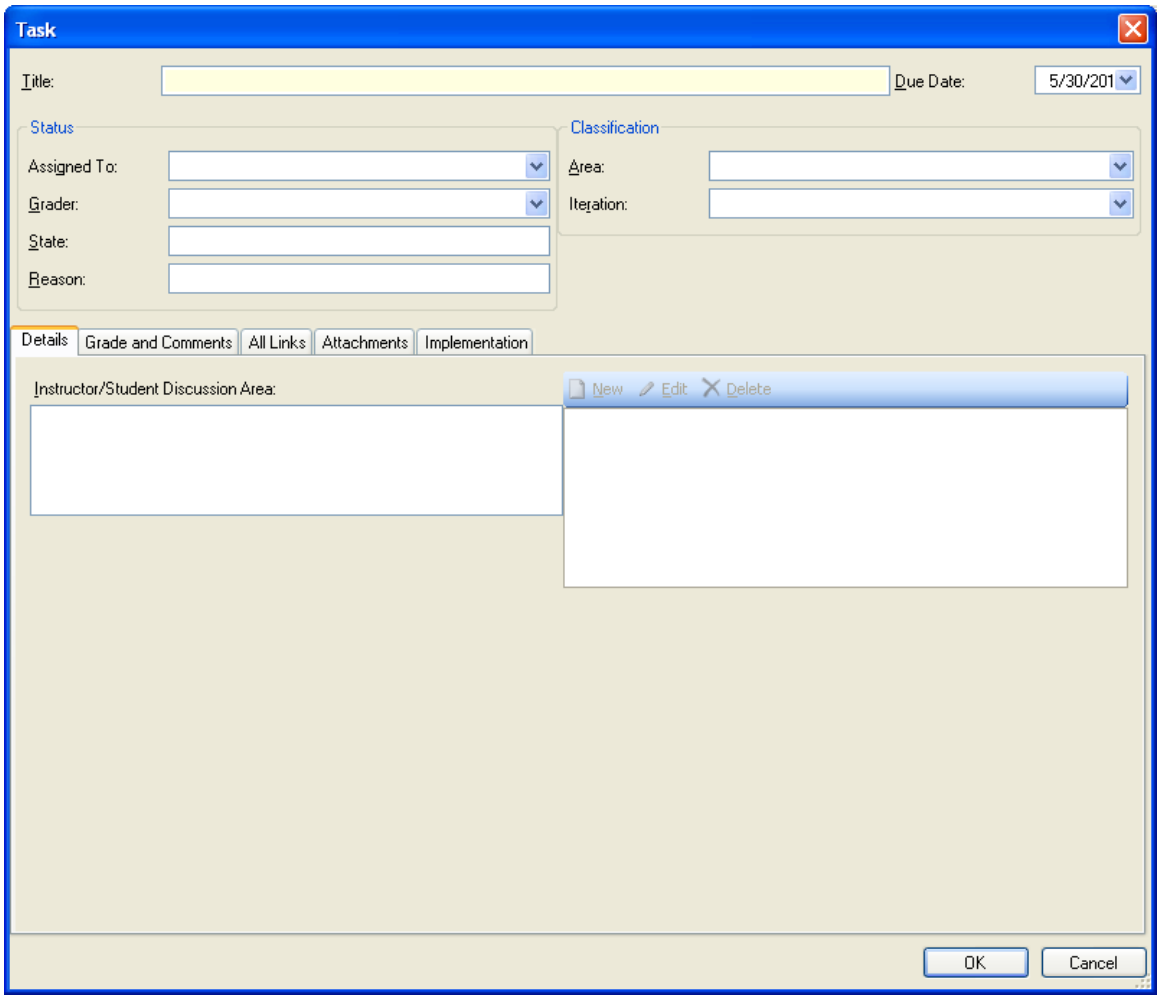


Appendix K.6 – As Planned Actor Diagram

Appendix L.1 – Workflows: Work Item Task



Appendix L.2 – Workflows: Form Layout



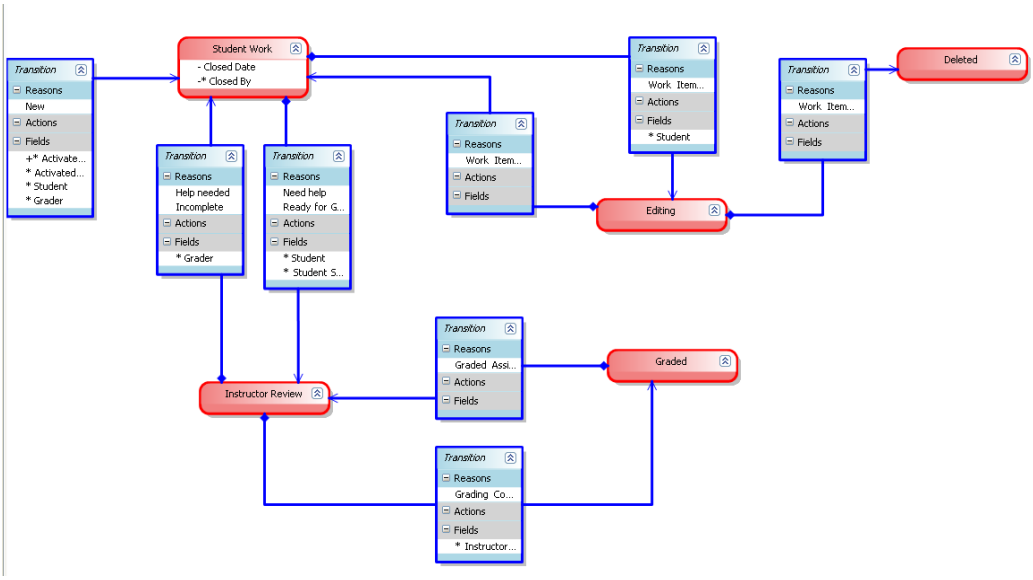
Appendix M.1 – Interview with Dr. Tom Janicki

June 2, 2011 at 2 pm

Agenda:

Discuss how the structure and workflows his MIS 413 class would be different from the MIS 216 class. The mini projects would be similar but there is the issue of when they start their client projects as you cannot assign a single task to two people.

MIS 216 Workflow



Open Ended Questions:

1. What is the structure of your MIS 413 class?

The server miscapstone is used with a folder for the MIS 413 classes.

   a. Specifically the folder structure on miscapstone.

Each student has their own folder on the server with permissions to it. This allows the students to have access to just their own folder and Dr. Janicki has access to all of them.

   b. The assignments

The semester starts out with three incremental assignments that Dr. Janicki calls mini project 1, 2, and 3. The students then have eight client checkpoints which include the tasks needed to be done for each checkpoint.

2. How are the mini project assignments given and submitted?

The mini project details are located as pdfs on Dr. Janicki's website. The student retrieves the assignment details and begins work on the assignment. When the assignment is due the student will do a network copy and paste to their folder on miscapstone. They are also required to send an e-mail to

Dr. Janicki with a URL that will allow him to click the link and run the students' work on the live server and not locally. This is done because when it is run locally it ignores permissions which is something that Dr. Janicki needs to be able to observe.

a. How is feedback given?

Feedback is given in one of three ways. The first way is when their grade is entered in Entropy Dr. Janicki will add any comments there. The second way is that Dr. Janicki will print the page, whether it is a form, code, database table, or the relationship diagram in the database. He will write comments on the page(s) and hand them back to the students. The last way is that if in the code the student is doing something that is obviously wrong or there is a much better solution, Dr. Janicki will insert pseudo code for how to do it and then comment it out. He then pushes the changed file to the student's folder on miscapstone and informs the student.

b. Are they required to use feedback to fix existing problems?

Absolutely. When students submit an assignment, feedback is typically returned to them by the next class. This is done because the assignments are incremental and any mistakes they made that are identified by Dr. Janicki are required to be fixed for the next submission. If a student fails to fix any of these problems they are penalized double.

3. How are client projects given and submitted?

Client projects are assigned the same way that the mini projects are assigned; the details are in a pdf that are located on Dr. Janicki's website. There is one additional part of the client projects which is a project plan. The client project team must create a document that defines there project plan; what work is going to be done, who is it going to be done by, when will it be done. This document is updated and resubmitted with each client submission.

a. How do they work on the same code?

The students do not work on the same code. When they start the client project the database is designed by the team. All work done in the team's database uses a naming convention where they add their initials to the beginning of the table, diagram, or stored procedure that they create. This allows Dr. Janicki to see what team member has done in the database.

When the team starts coding they work separately to create a master page. When the team meets with Dr. Janicki they review both master pages and decide as a team which one to use. The master page that is chosen is then copied into the other team member's folder and used for the rest of the client project. At this point all the work that is done on the client project is done separately with each team member copying their work into their individual folder on the miscapstone server. A separate e-mail is sent to Dr. Janicki from both team members with a link to be able to run each of their pages along with an attached update project scope document.

The team members are graded separately and for each checkpoint the team members project plan is used to grade by looking at one they did versus what they said they would do. If a team member fails to do something that was in the project plan but identified something that they overlooked that needed to be done before what they said would be done; then Dr. Janicki will look to see if they did this and only a few points are taken off. If they fail to do what needed to be done for what they said would be done, then it is viewed as they did not meet the project plan and are penalized in full.

For the final submission of team member's project their files are merged together. This works because the team members work in different folders under the project in different areas. Dr. Janicki uses the team member's project plan to grade them based on what they said they would do versus what they did.

b. Are they required to perform peer review the other partners work before submission?

No peer review is done due to the fact that students work separately and are graded individually.

4. How do you see the workflow for your MIS 413 class structured?

The mini projects could use almost the same workflow as the MIS 216 class with one exception. Dr. Janicki wants the ability for students to submit work item tasks for help because students will tend to abuse this ability. This could lead to situations where the student submits a task for help for one line of code and will stop work until they receive a response. With the ability to abuse this feature of the workflow students could do this for every line of code which would overwhelm the instructor.

For the client projects, the assignments where team members are working in Visual Studio on code the work item tasks could be used. For the other client projects where they are working on the database the work item tasks would not be useful.

Appendix M.2 – Interview with Dr. Douglas Kline

January 19, 2011 at 1:15pm

How are assignments given to the students?

Typically a case from the text book is assigned and a write up of the assignment is done by Dr. Kline referencing the case in the book. The write up includes additional requirements, things to take note of, method signatures, basic coding standards, definitions, and submission instructions.

What programming tools do the students use?

Primarily Visual Studio 2008, going to upgrade to Visual Studio 2010 in fall of 2011.

What programming tools do you, the faculty, use?

Visual Studio

How do students submit their work?

Students are required to zip their project folder and upload that zip file to Entropy. This is a shared folder available to Dr. Kline on the network. When Dr. Kline is ready to grade, he goes to the shared network folder and copies all of the students zip files to a folder on his desktop. He then goes through and manually unzips each student's files putting them in a folder abc1234_MIS216_001. Once all of the files have been unzipped, he uses a program that he wrote called ProjectPrinter. This program lets him point at the project folder containing the folders of all the students' projects. The program then recursively finds all of the vb files and prints them. The last thing that he must do before he is ready to grade is to create a checklist. This checklist is printed out for each of the students. Once he has the students zip files unzipped, the vb files printed, and the checklist, he is then ready to sit down and grade.

Normally he doesn't open up the student's files in VS to see if they will run. The only time this occurs is if there is a question about what a student did, or a question as to what would happen when it runs. The grading is done using the printouts using the checklist as a guide. Comments are written on the printouts of the vb files. The whole process takes roughly 2-3 hours. Once the preliminary tasks are completed, the grading itself takes only about an hour to an hour and a half. From the due date of the assignment, it takes about 1-2 weeks to get the checklist and commented vb code back to the students. There are late submissions, normally one day late unless there are extenuating circumstances.

Do students do any group work?

Sometimes on the last big project Dr. Kline will give the students the option of pairing up. The students will work on the same code together using the pair programming technique of using one computer. For thirty minutes one person will code while the other observes and then switches. The students are not graded individually, but instead are given an overall grade. The argument for the overall grade instead of grading individually is first they are not working in parallel but on the same code. Second, pairing up is not a requirement so if a pair member is not doing the work, the other member can choose to continue working on the assignment alone.

Appendix M.3 – Interview with Kevin Matthews

May 25, 2011 at 12:30 pm

How are assignments given to the students?

The assignment details are created and saved as pdfs on Mr. Matthew's website.

What programming tools do the students use?

Primarily Visual Studio 2008, going to upgrade to Visual Studio 2010 in fall of 2011.

What programming tools do you, the faculty, use?

Visual Studio 2008

How do students submit their work and how does grading and feedback work?

Students are required to zip their project folder and upload that zip file to Entropy. This is a shared folder available to Mr. Matthews on the network. When Mr. Matthews is ready to grade, he goes to the shared network folder and copies all of the students zip files to a folder on his desktop. He then goes through and manually unzips each student's files putting them in a folder abc1234_MIS216_001. Each student's project is opened in Visual Studio and a build/run is done. Using the grading checklist Mr. Matthews goes through and reviews the layout and properties of each of the students' form and then goes through and reviews their code. Once the grading is done the grading checklist with the students' name and any comments that were added in the margins are printed and handed back to the students.

The whole process of grading takes on average about hour to an hour and a half for the first couple of assignments. Once the assignments get more complicated the grading process can take two to two and a half hours. The assignments at this level are not incremental so there is not the need to get results returned to students by the next class. Because of this feedback usually takes about a week to return to students.

Do students do any group work?

At this introductory level class no student work is done.

Appendix M.4 – Interview with Dr. Devon Simmonds

June 2, 2011 at 1:30pm

Agenda:

Discuss other Software Engineering tools that would be applicable to my project. I am using Team Foundation Server which provides full Configuration Management, source control, process templates, reporting, enforcement policies, work items, etc.

Open Ended:

He discussed with me Software Engineering tools and if I am proposing to use an environment like Team Foundation Server that I need to look at the other ones out there. The reason for doing this is when I do my defense, if I am asked questions about why this environment should be used I can compare it to the other options out there and show how it is better. He also gave me documents and links to sources that went over in detail configuration management, source control, and other tools.

For testing Dr. Simmonds and I discussed using case based testing to test the success of my project from both the student and Instructor side. By doing this Dr. Simmonds explained that I would be able to show what parts of my project were successful and which parts needed further work or refinement. The biggest piece of advice that Dr. Simmonds gave me was that any time I identified a problem to document it immediately. He discussed that this is a huge part of my final defense and I need to be able to accurately show the problems I encountered and the lessons I learned from them.

Appendix N

UNCW/CSB/ISOM is concerned with your privacy. All answers are confidential and are to be used only in the ISOM department.

On a scale of 1 to 5 with 1 being the LEAST difficult and 5 being the MOST difficult;

1. How would you rate the overall difficulty of connecting to the Team Foundation Server with Visual Studio?

   1 – Very Low, 2 – Low, 3 – Average, 4 – High, 5 – Very High

2. How would you rate the overall difficulty of mapping source control to a local drive?

   1 – Very Low, 2 – Low, 3 – Average, 4 – High, 5 – Very High

3. How would you rate the overall difficulty of checking project files in and out from source control?

   1 – Very Low, 2 – Low, 3 – Average, 4 – High, 5 – Very High

4. How would you rate the overall difficulty of using the work item tasks to complete assignments?

   1 – Very Low, 2 – Low, 3 – Average, 4 – High, 5 – Very High

How well do you understand the following concepts on a scale of 1 to 5 with 1 being you have no understanding and 5 being you understand it very well?

5. How well do you understand the concept of a central location for your work?

   1 – No understanding, 2 – A little, 3 – average, 4 – well, 5 – very well

6. How well do you understand the concept of source control?

   1 – No understanding, 2 – A little, 3 – average, 4 – well, 5 – very well

7. How well do you understand the concepts of checking in and out of source control?

   1 – No understanding, 2 – A little, 3 – average, 4 – well, 5 – very well

8. How well do you understand how to use work item tasks for your assignments?

   1 – No understanding, 2 – A little, 3 – average, 4 – well, 5 – very well

How useful on a scale of 1 to 4 with 1 being the LEAST useful and 4 being the MOST useful would you consider the following to be?

9. How useful do you consider version control to be?

   1 – No use, 2 – Somewhat useful, 3 – Useful, 4 – Very Useful

10. How useful did you find the user guides for working with Visual Studio and TFS?

    1 – No use, 2 – Somewhat useful, 3 – Useful, 4 – Very Useful

11. How useful was the feedback provided by your work item tasks?

    1 – No use, 2 – Somewhat useful, 3 – Useful, 4 – Very Useful

If you have any additional comments, suggestions, or complaints please feel free to include them in the section below.

Additional comments:

Appendix O

Survey Results

On a scale of 1 to 5 with 1 being the LEAST difficult and 5 being the MOST difficult;				
How would you rate the overall difficulty of connecting to the Team Foundation Server with Visual Studio?				
Very low	Low	Average	High	Very High
4	12	4	0	0
How would you rate the overall difficulty of mapping source control to a local drive?				
Very low	Low	Average	High	Very High
3	13	4	0	0
How would you rate the overall difficulty of checking project files in and out from source control?				
Very low	Low	Average	High	Very High
5	12	2	1	0
How would you rate the overall difficulty of using the work item tasks to complete assignments?				
Very low	Low	Average	High	Very High
4	14	2	0	0
How well do you understand the following concepts on a scale of 1 to 5 with 1 being you have no understanding and 5 being you understand it very well?				
How well do you understand the concept of a central location for your work?				
No Understanding	A little	Average	Well	Very well
0	1	1	10	8
How well do you understand the concept of source control?				
0	5	3	8	4
How well do you understand the concepts of checking in and out of source control?				
0	1	6	9	4
How well do you understand how to use work item tasks for your assignments?				
0	2	5	6	7
How useful on a scale of 1 to 4 with 1 being the LEAST useful and 4 being the MOST useful would you consider the following to be?				
How useful do you consider version control to be?				
No use	Somewhat useful	Useful	Very useful	
2	8	6	4	
How useful did you find the user guides for working with Visual Studio and TFS?				
0	1	7	12	
How useful was the feedback provided by your work item tasks?				
1	3	14	2	

Appendix P

Class Setup

1. Create new team project for Agile Development, no SharePoint, empty folder.
2. From the team project, select team project settings.
    a. Select group memberships
        i. Create students
        ii. Create instructor
    b. Find the students group and go to properties
        i. Add all students
    c. Find the instructor group and go to properties
        i. Add instructor
3. From the team project, select the team project settings
    a. Select security
        i. Add student group
            1. Deny everything except view project level
        ii. Add instructor to group
            1. Allow all except delete team project
4. From the team project, select Areas and Iterations.
    a. Add child areas for each student using the student id.
    b. Select the security on each area
        i. Add windows user group
        ii. Browse student
        iii. Deny all except edit and view work items
    c. Add Iteration for each assignment with name
        i. No security changes
5. Source control folder
    a. Map source control at root to the instructor local work space.
    b. New folder for each student named using the student ids.
    c. Check in at root
    d. Right click and select Properties on each student's folder, and then security.
        i. Add Windows user
        ii. Student Id
        iii. Allow first 4, #6 and #8
        iv. Deny everything else
6. If not already installed, install TFS power tools.
    a. Once power tools is installed, go to Tools > Process editor > Work Item Types > Export WIT
    b. Select a previous class with correct settings
    c. Choose task as work item type
    d. Browse to the store location

    e. Yes, include global list definition
    f. Go to Tools > process editor > work item types > import WIT
    g. Brows to pickup tnc file from the export from 6d.
    h. Highlight new team project as the destination
7. Right click team project in the team explorer tab.
    a. Team project settings > source control
    b. Check-in policy
    c. Add
        i. Changeset comments policy
        ii. Work items policy

Appendix Q

Student User Guides

UNIVERSITY OF NORTH CAROLINA WILMINGTON

MIS 216 AND MIS 316

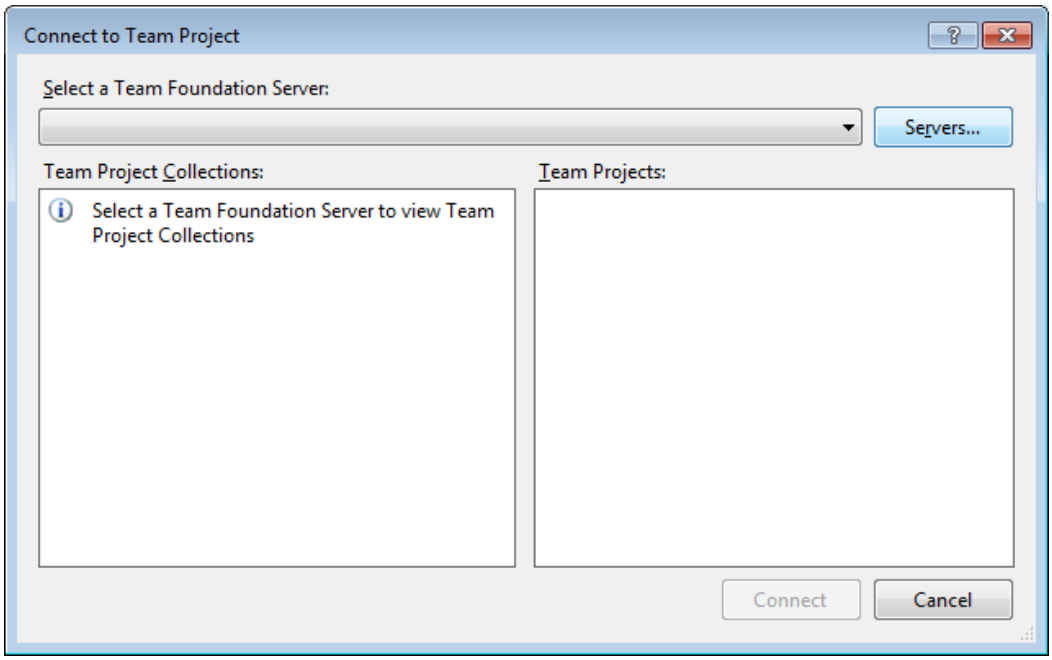
CONNECTING TO TEAM FOUNDATION SERVER (TFS)

Team Foundation Server (TFS) is a project management and source control tool that can be utilized inside many Microsoft products including Visual Studio 2010. In order to access TFS, you MUST USE THE *PREMIUM* VERSION (from me) OF VISUAL STUDIO 2010 **NOT** THE *EXPRESS* VERSION (with the book or from Microsoft site). Use the following step-by-step directions to help you connect to our Team Foundation Server.

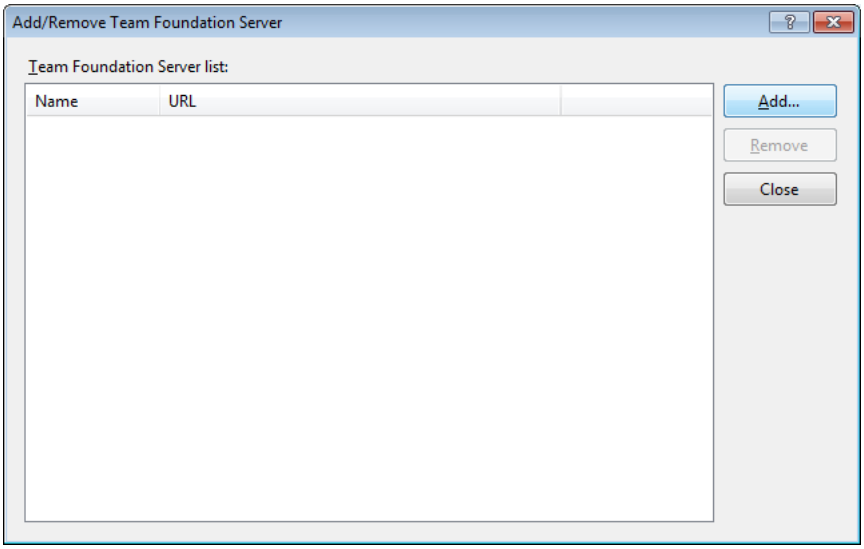
1. Open Visual Studios by locating the program shortcut under *Start > All Programs > Microsoft Visual Studio 2010*.

2. On the Start Page, locate and click the link that says *Connect to Team Foundation Server* in the top left as shown here.

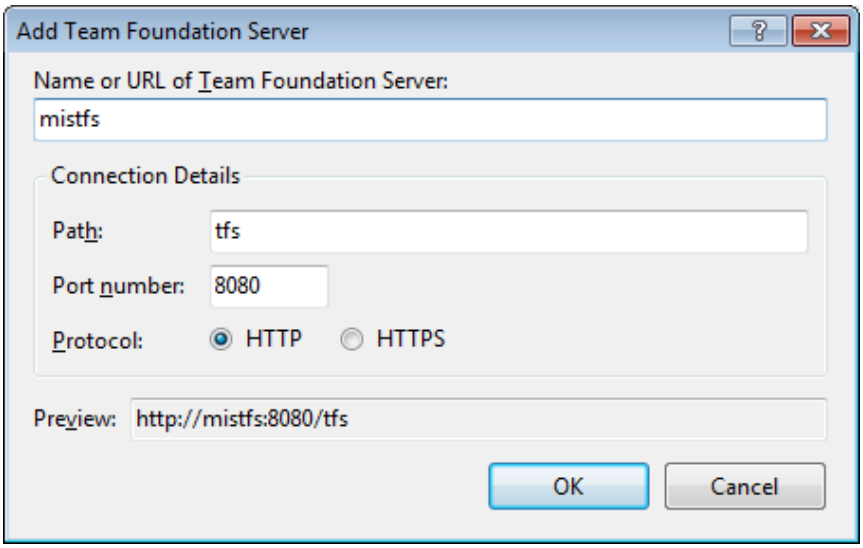
3. You should now see the *Connect to Team Project* window. Click the *Servers...* button on the right of the dropdown box.



4. You should now see the *Add/Remove Team Foundation Server* window. To continue, click the *Add...* button.



5. On the *Add Team Foundation Server* window, enter the server details to connect to the **mistfs** Team Foundation Server as shown below (you just have to type the name):



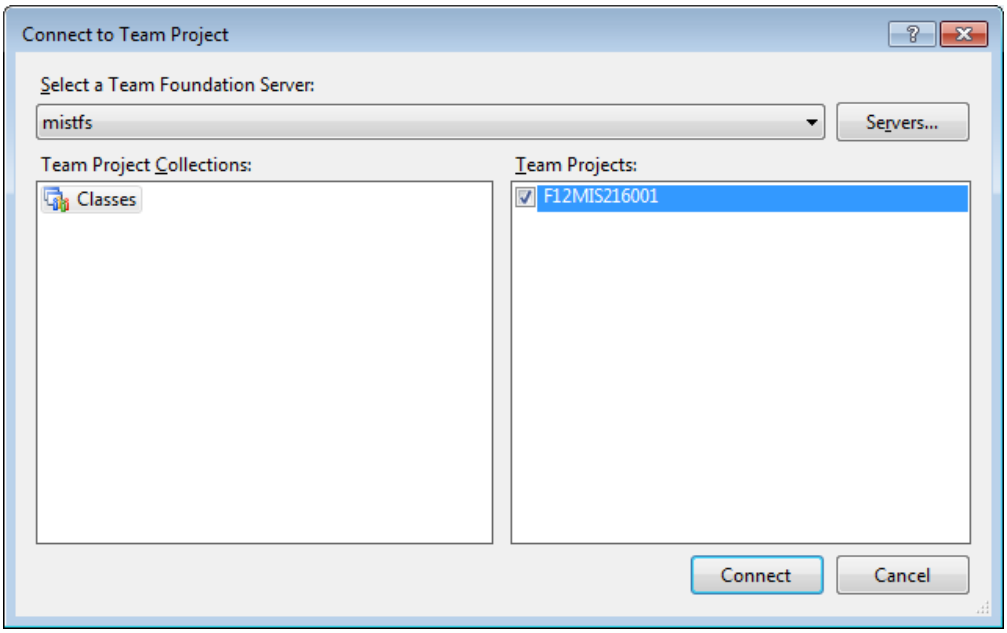
6. Click the *OK* button.

7. You should now see the server listed on the *Add/Remove Team Foundation Server* window. You can click the *Close* button.

8. Now that the server has been added, it should be visible in the dropdown at the top of the *Connect to Team Project* window. If not, click the down arrow on the dropdown box and select it from the list.

9. On the left under *Team Project Collections:* you should see a *Classes* option. Click on *Classes*.

10. The right side labeled *Team Projects:* should now have options to select. You should see an **F12MIS216001** or **F12MIS316001** option (you may also see previous semesters if you were in another course. Check the box beside this option to select this course.



11. Click the *Connect* button.

12. You are now connected to our class's *Team Project*. **F12MIS216001** or **F12MIS316001** should now be visible in the *Team Explorer* tab. This tab should be on the right of your screen with the *Solution Explorer* tab. If you do not see it, click *View > Team Explorer* in the top menu bar.

UNIVERSITY OF NORTH CAROLINA WILMINGTON

MIS 216 AND MIS 316

GETTING A PREVIOUS VERSION OF A PROJECT

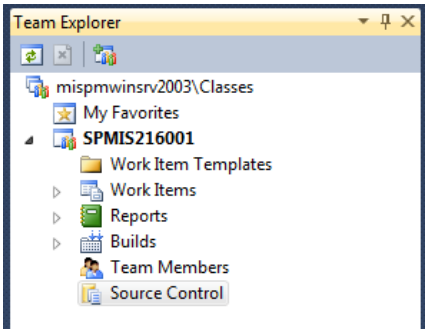
Team Foundation Server (TFS) allows you to get the latest version or a previous version of a project under *Source Control*. Each time you *Check In* a project, a "snapshot" of your code is saved. TFS calls this "snapshot" a **Changeset**. This allows you to easily "roll back" to that version of code if needed. Follow these instructions to see the available change sets and roll back to a previous changeset.

13. Open Visual Studios by locating the program shortcut under *Start > All Programs > Microsoft Visual Studio 2010*.

14. Make sure you are properly connected to TFS. You should be connected to the **mistfs** server and have the appropriate Team Project loaded. You may need to reconnect to the server. There are separate instructions for this on the course website.

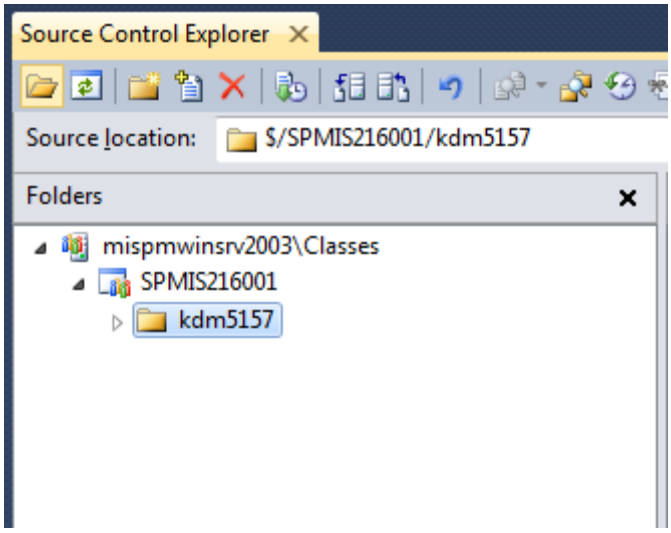
15. Once you are connected to our class's *Team Project*, it should be visible in the *Team Explorer* tab. This tab should be on the right of your screen with the *Solution Explorer* tab. If you do not see it, click *View > Team Explorer* in the top menu bar.

16. In the *Team Explorer* window, double click the *Source Control* option. This will display the folders you have access to on the Team Foundation Server.



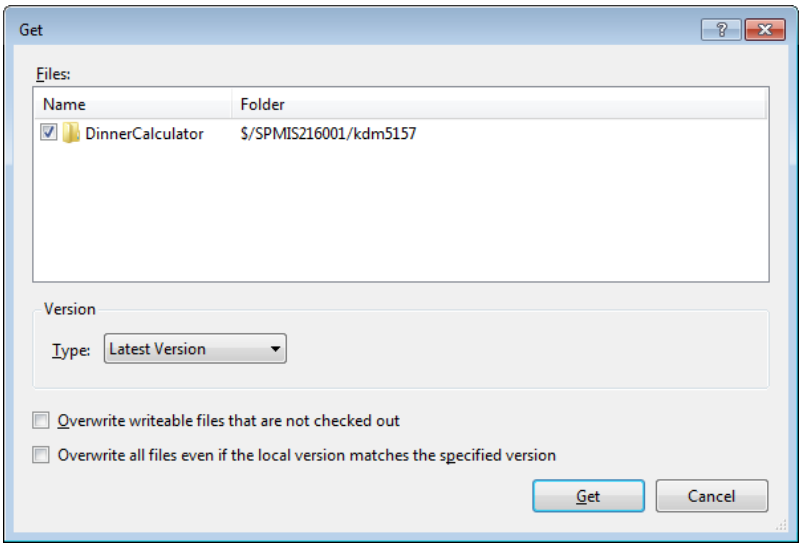
17. In the *Source Control Explorer*, you will see the class's *Team Project* on the left. Double-click the name to expand this option.

18. You should see a folder with your UNCW email ID on the left under the Team Project name. This folder is unique based on your login and can only be seen by your account.

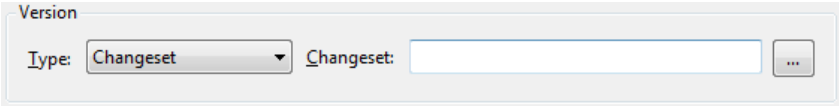


19. The following can be done for ANY folder in your *Source Control*. Typically each folder represents a different project/assignment for this course.

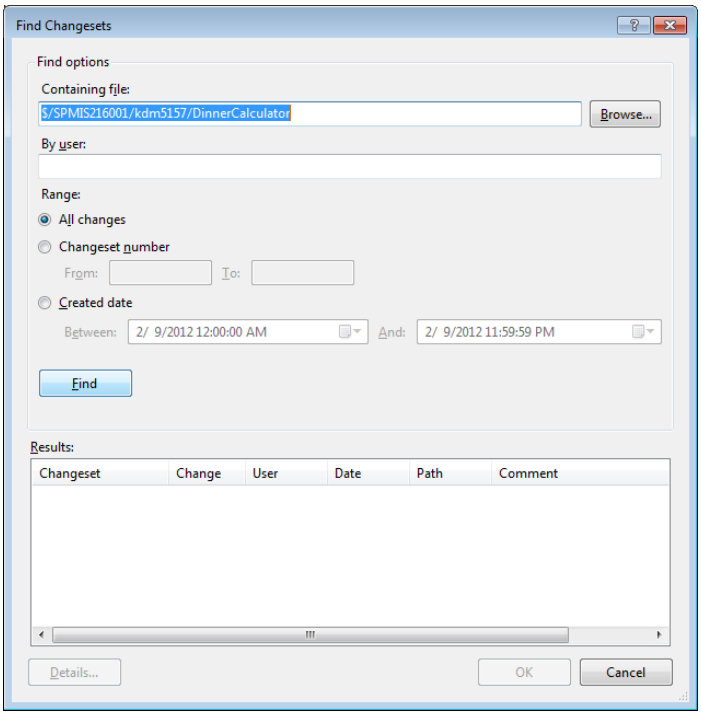
20. Right click on the folder you wish to roll back. Select the option *Get Specific Version...*, you should see the *Get* window.



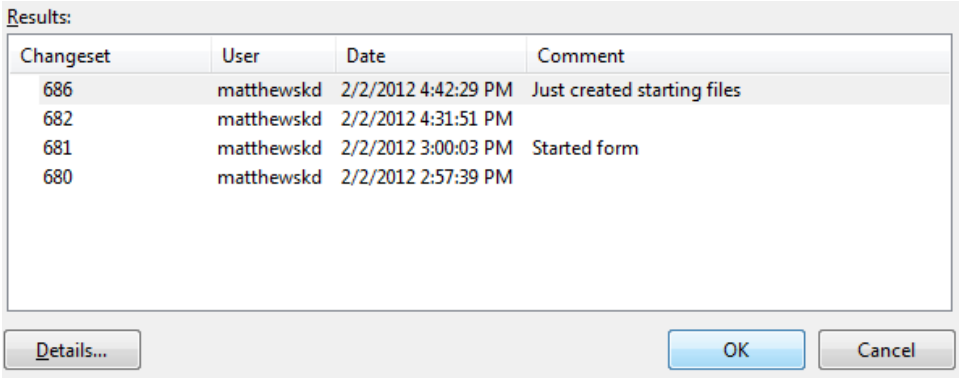
21. By default, the *Type* selection is *Latest Version*. Click the dropdown and choose the *Changeset* option. You should see a textbox with a small button beside it. Click the … button to search for the available changesets.



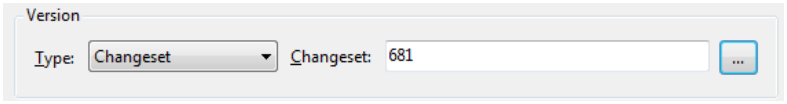
22. In the *Find Changesets* window, you should notice the folder you selected as the *Containing File* option. Click the *Find* button.



23. You should now see all changesets for this project. To select one, highlight it and click the *OK* button. This will download and replace the files you currently have. One of the great things is that your files are NOT LOST. As long as you checked in the files, the changesets are available. You can jump to previous files, back to future files, and so on and so on. All changesets are continually available.



24. Back on the *Get* window, you should see the selected changeset number listed. Click the *Get* button to finish this process of “rolling back”.


Version
Type:
Changeset
Changeset:
681
...

UNIVERSITY OF NORTH CAROLINA WILMINGTON

MIS 216 AND MIS 316

MAPPING SOURCE CONTROL IN TFS

(THIS SHOULD ALWAYS BE THE FIRST THING YOU DO AFTER CONNECTING TO TFS)

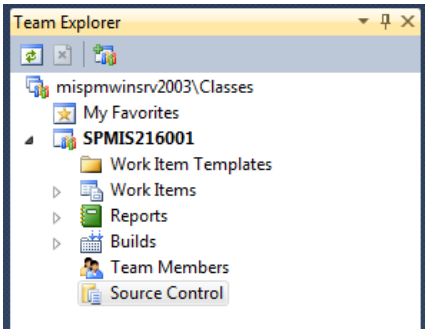
Team Foundation Server (TFS) allows you to access remote files & folders under what is called *Source Control*. Follow these steps to ensure you are using Source Control and mapped to your hard drive.

25. Open Visual Studios by locating the program shortcut under *Start > All Programs > Microsoft Visual Studio 2010*.

26. Make sure you are properly connected to TFS. You should be connected to the **mistfs** server and have the appropriate Team Project loaded. You may need to reconnect to the server. There are separate instructions for this on the course website.

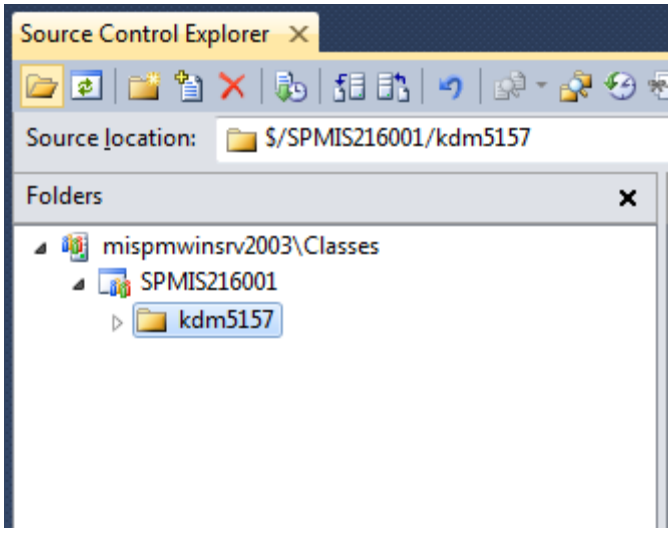
27. Once you are connected to our class's *Team Project*, it should be visible in the *Team Explorer* tab. This tab should be on the right of your screen with the *Solution Explorer* tab. If you do not see it, click *View > Team Explorer* in the top menu bar.

28. In the *Team Explorer* window, double click the *Source Control* option. This will display the folders you have access to on the Team Foundation Server.



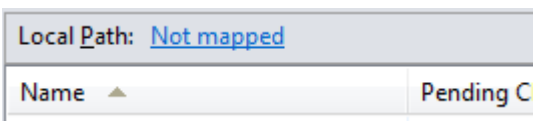
29. In the *Source Control Explorer*, you will see the class's *Team Project* n the left. Double-click the name to expand this option.

30. You should see a folder with your UNCW email ID on the left under the Team Project name. This folder is unique based on your login and can only be seen by your account.

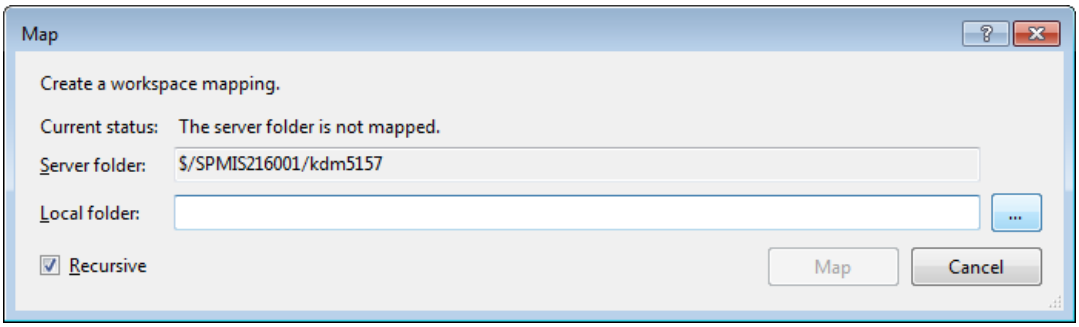


31. This folder is a NETWORK folder; it is NOT local to your hard drive. In order to work with this folder, you need to "map" the folder to have an identical folder on your local hard drive. Unfortunately, you will need to map this each time you use Visual Studios in the lab. I suggest mapping to a location on your Timmy drive and be consistent in that when using Visual Studios.

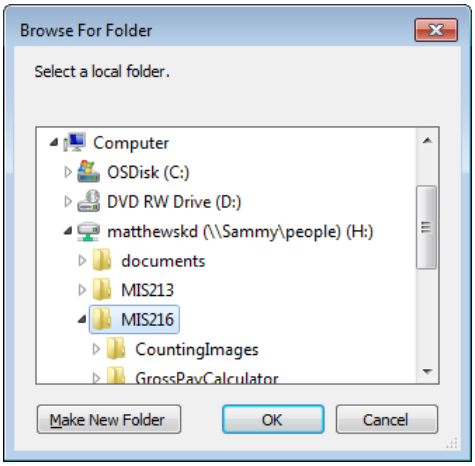
32. To the left the *Folders* in the *Source Control Explorer* you should see a screen that has the label *Local Path: Not mapped* across the top. Click the words *Not mapped* to open the screen to choose your local path.



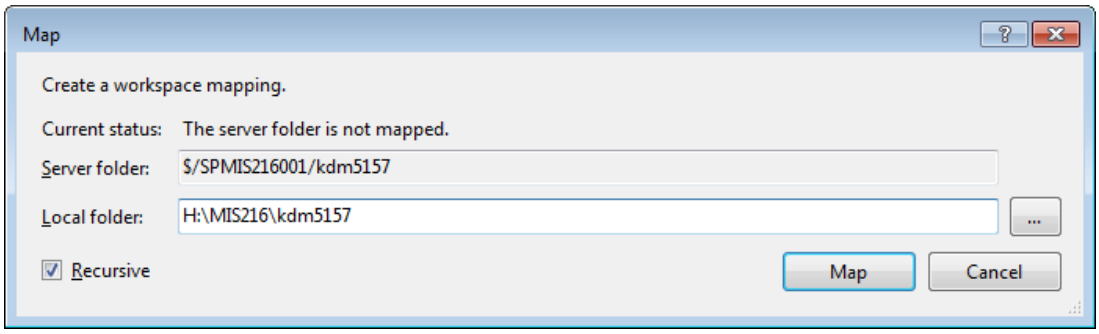
33. From the *Map* window, click the ... button next to the *Local folder* box.



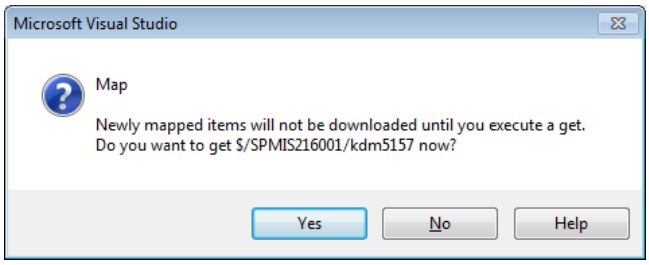
34. From the *Browse For Folder* window, find your *Timmy* drive and select a folder you wish to map your files to.



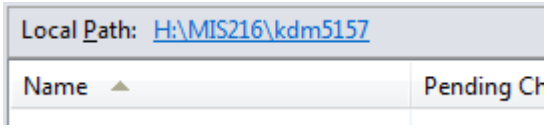
35. The *Local Folder* path should now show the path to your *Timmy* drive. If all of the information is correct, click the *Map* button.



36. A new popup will appear that lets you know that “Newly mapped items will not be downloaded until you execute a get. Do you want to get…” Select YES.



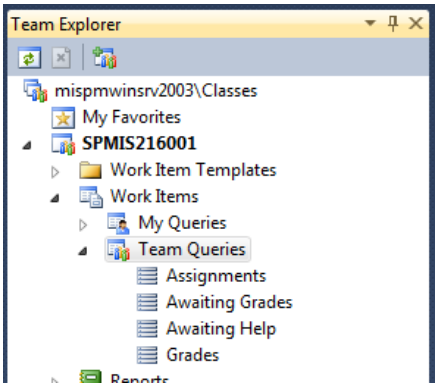
37. Now you should see the directory you selected (your *Timmy*) next to the *Local Path* label at the top. This is where your files will be created locally.



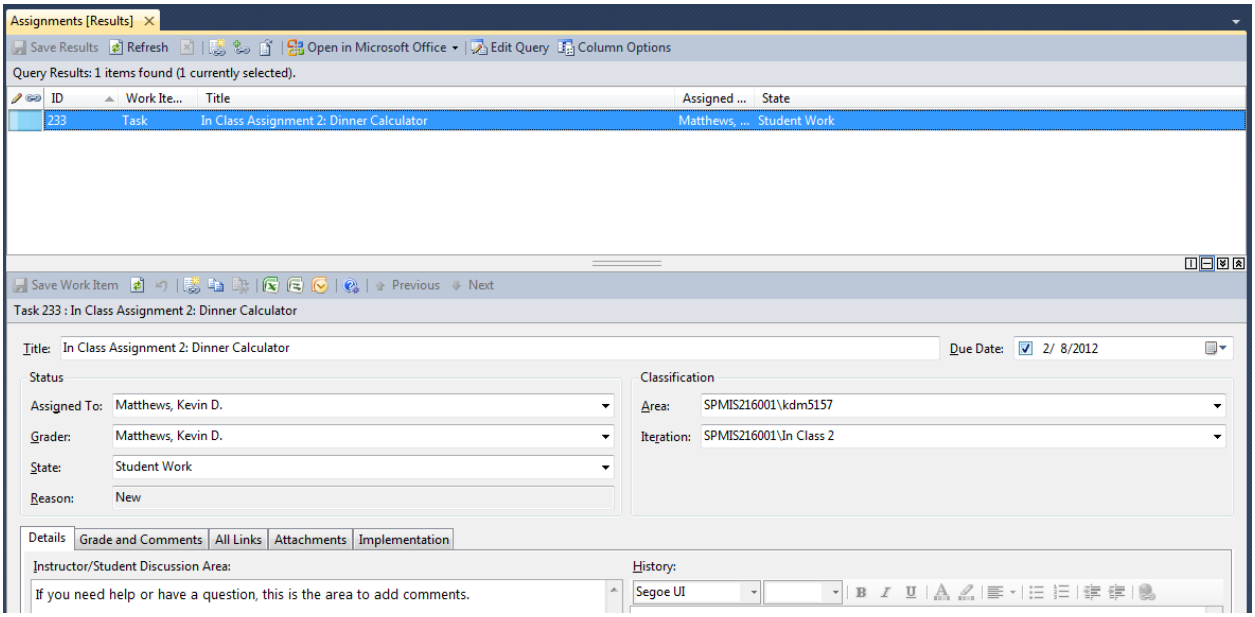
SUBMITTING WORK ITEMS FOR GRADING IN TFS

Before you submit a work item for any reason, make sure that you CHECK IN the latest version of your code.

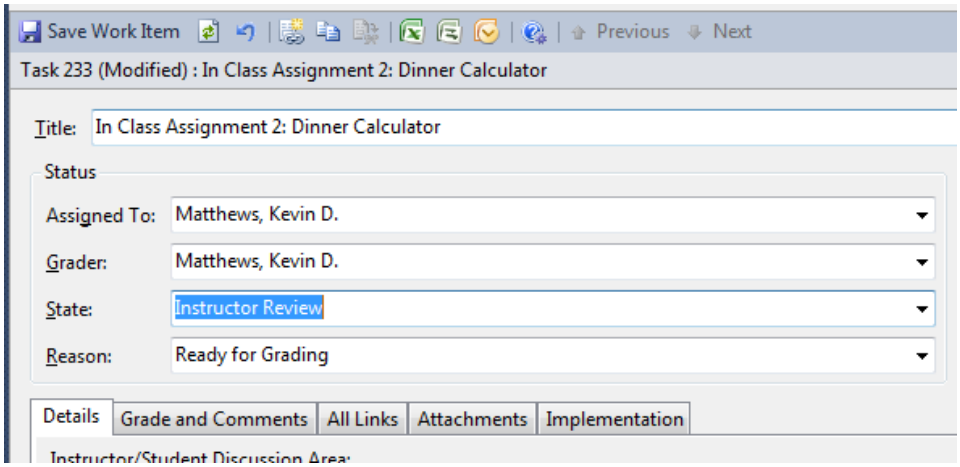
38. In the *Team Explorer* tab on the left, double click the option labeled *Work Items*. This will display *My Queries* and *Team Queries*. Double-click *Team Queries* to open these options. Double-click *Assignments* and a list of current tasks should display in the left panel.



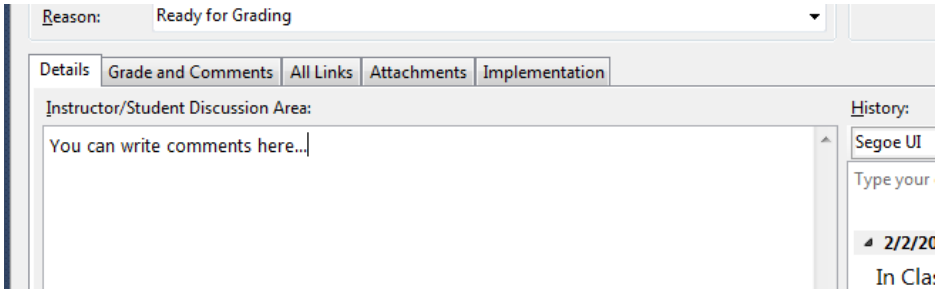
39. Select the task you wish to submit for grading. The details should display in the bottom panel.



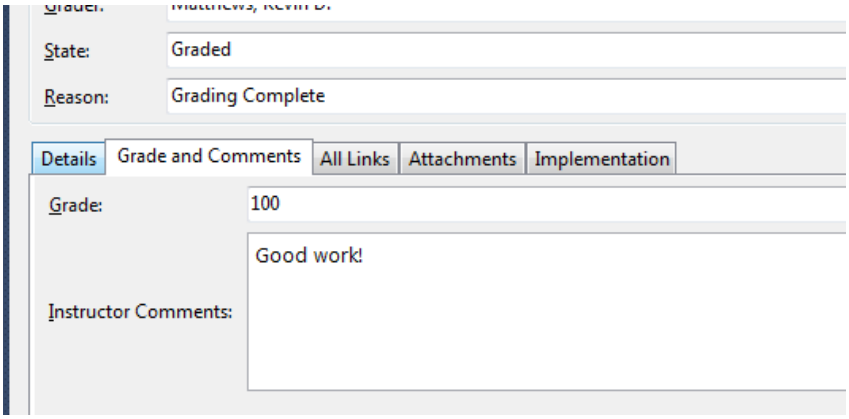
40. To submit your task for grading, click the dropdown next to the *State* option. Change the option to *Instructor Review*. Notice that the *Reason* option changes to *Ready for Grading*.



41. Add any comments in the *Details* window and then click the *Save Work Item* icon.



42. While awaiting a grade, the assignment will be listed under *Awaiting Grades* in the *Team Explorer* tab on the right. Once a grade has been given, the assignment will be listed under *Grades*. You can select a task and view the *Grade and Comments* tab in the work item details to see what feedback you received.


State:
Graded
Reason:
Grading Complete
Details
Grade and Comments
All Links
Attachments
Implementation
Grade:
100
Good work!
Instructor Comments:

UNIVERSITY OF NORTH CAROLINA WILMINGTON

MIS 216 AND MIS 316

WORKING WITH WORK ITEMS IN TFS

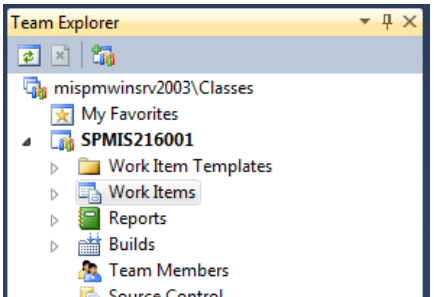
Team Foundation Server (TFS) allows me to assign *Work Items* to you for each programming assignment we will complete. This guide will help you understand how to work with work items.

43. Open Visual Studio by locating the program shortcut under *Start > All Programs > Microsoft Visual Studio 2010*.

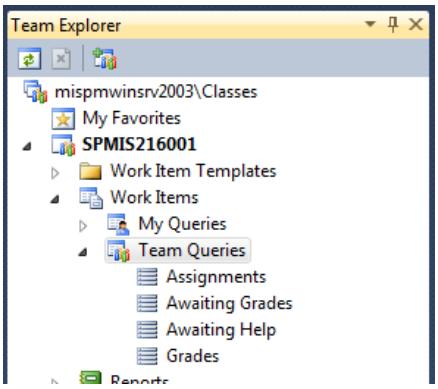
44. Make sure you are properly connected to TFS. You should be connected to the **mistfs** server and have the appropriate Team Project loaded. You may need to reconnect to the server. There are separate instructions for this on the course website.

45. Once you are connected to our class's *Team Project*, it should be visible in the *Team Explorer* tab. This tab should be on the right of your screen with the *Solution Explorer* tab. If you do not see it, click *View > Team Explorer* in the top menu bar.

46. In the *Team Explorer* window, double click the *Source Control* option. This will display the folders you have access to on the Team Foundation Server.



47. Make sure that the appropriate class option on the left is mapped to your *Timmy* drive. There are separate instructions for this on the course website.

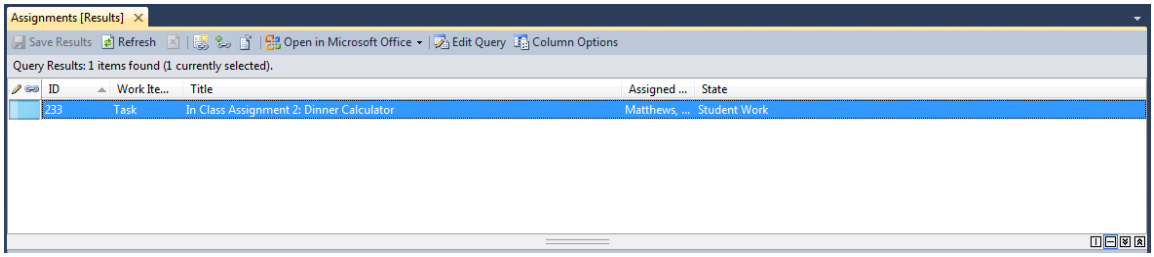


48. Now that you have connected to the server and mapped your local folder, you are ready to begin working. To get your assignments, double click the option labeled *Work Items* in the *Team Explorer* tab on the left. This will display *My Queries* and *Team Queries*. Double-click *Team Queries* to open these options.

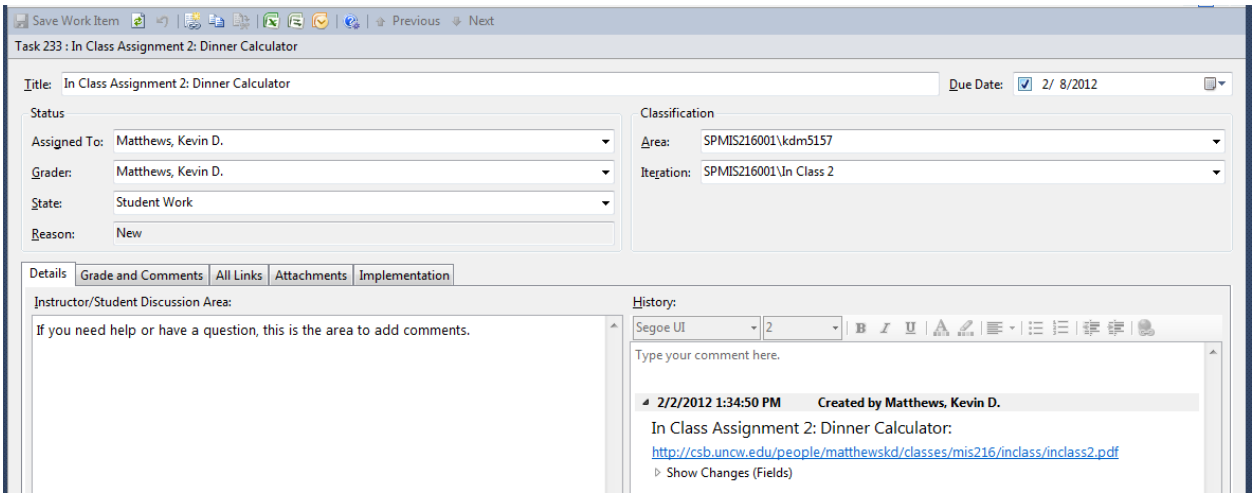
49. You should see options for *Assignments*, *Awaiting Grades*, *Awaiting Help*, and *Grades* as options. These options will show assignments at certain stages while we work.

50. Double-click *Assignments* and a list of current assignments should display in the left panel.

51. The top under *Query Results* you should see a *Task* that represents a programming assignment that you need to complete.



52. When you select a task, the details should display on the bottom half of the screen. This is the programming assignment that you need to complete. All links and information will be supplied in these details. Opening links will open in another tab within Visual Studio. A great benefit of using source control is that everything you need is all within Visual Studio – code, assignment details, and PDF directions. Also notice that the tasks is *Assigned To* you, and the *Area* should be your email ID. The starting *State* for all assignments will be *Student Work*. You can also see the *Due Date* of the assignment here.


Save Work Item
Previous
Next
Task 233 : In Class Assignment 2: Dinner Calculator
Title: In Class Assignment 2: Dinner Calculator
Due Date: 2/ 8/2012
Status
Assigned To: Matthews, Kevin D.
Grader: Matthews, Kevin D.
State: Student Work
Reason: New
Classification
Area: SPMIS216001\kdm5157
Iteration: SPMIS216001\In Class 2
Details
Grade and Comments
All Links
Attachments
Implementation
Instructor/Student Discussion Area:
If you need help or have a question, this is the area to add comments.
History:
Segoe UI
2
Type your comment here.
2/2/2012 1:34:50 PM Created by Matthews, Kevin D.
In Class Assignment 2: Dinner Calculator:
http://csb.uncw.edu/people/matthewskd/classes/mis216/inclass/inclass2.pdf
Show Changes (Fields)

Appendix R

Interview with Mr. Matthews

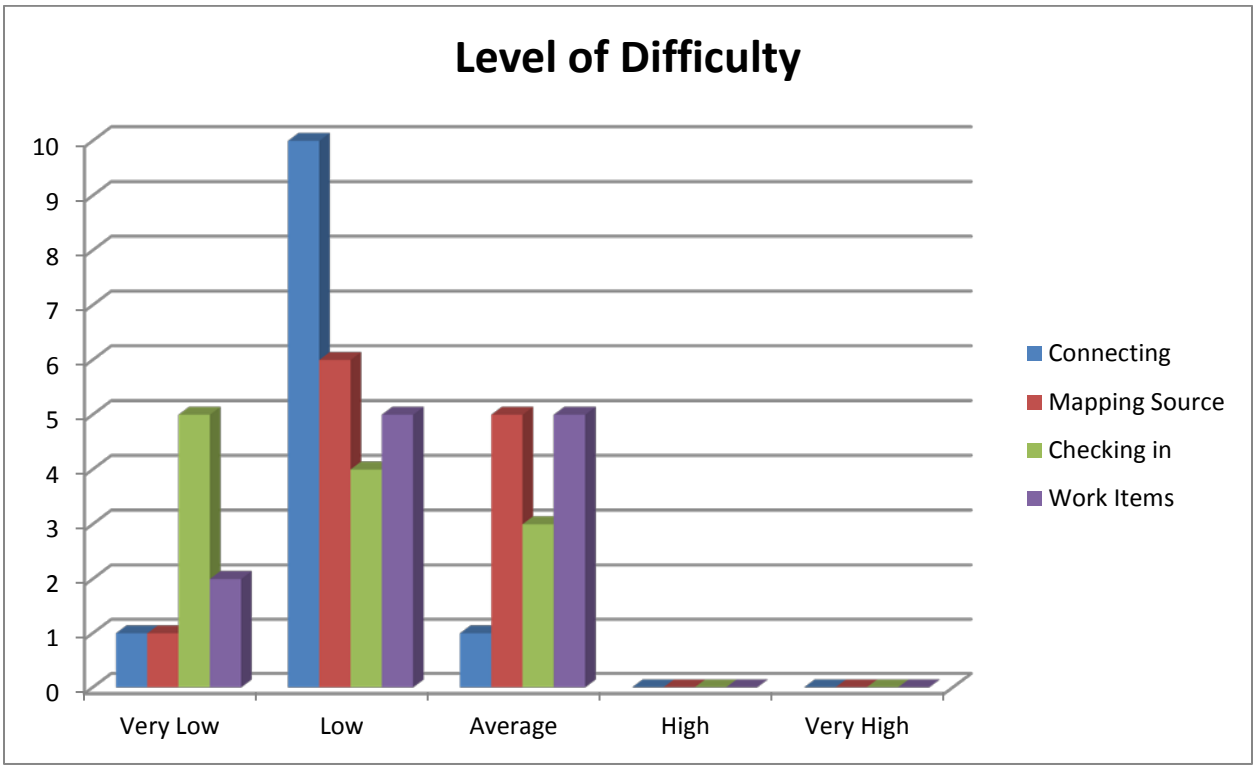
- How much time did it take you to set up a class in TFS?
  - One hour
- On average, how many students?
  - 35
- How much time did it take you on grading before?
  - Three hours
- How much time did it take you on grading using PeTE?
  - Hour and a half
- Were you able to provide more, substantial feedback?
  - Yes, there wasn't as much downtown between switching between students' projects.
- In your opinion, we the students able to understand the necessary concepts?
  - Yes, most students were able to understand how to do everything after a couple times of doing it. There were on average about three students per class that did not understand. However, these were typically the students that did not come to class.
- What would you change?
  - Assigning a project would require creating 35 separate work items and assigning them to the student and changing the area on the work item to the student's area. An automated way of doing this would be nice.
  - Students would often have multiple projects for the same assignment named slightly different. It would be nice if he could open the solution from the work item or at least know which solution was the one they checked in for grading.
- Did you continue to use this after your summer class?
  - Yes, he used it every semester after the pilot.
- Would you recommend this to the next instructor?
  - Yes, definitely.

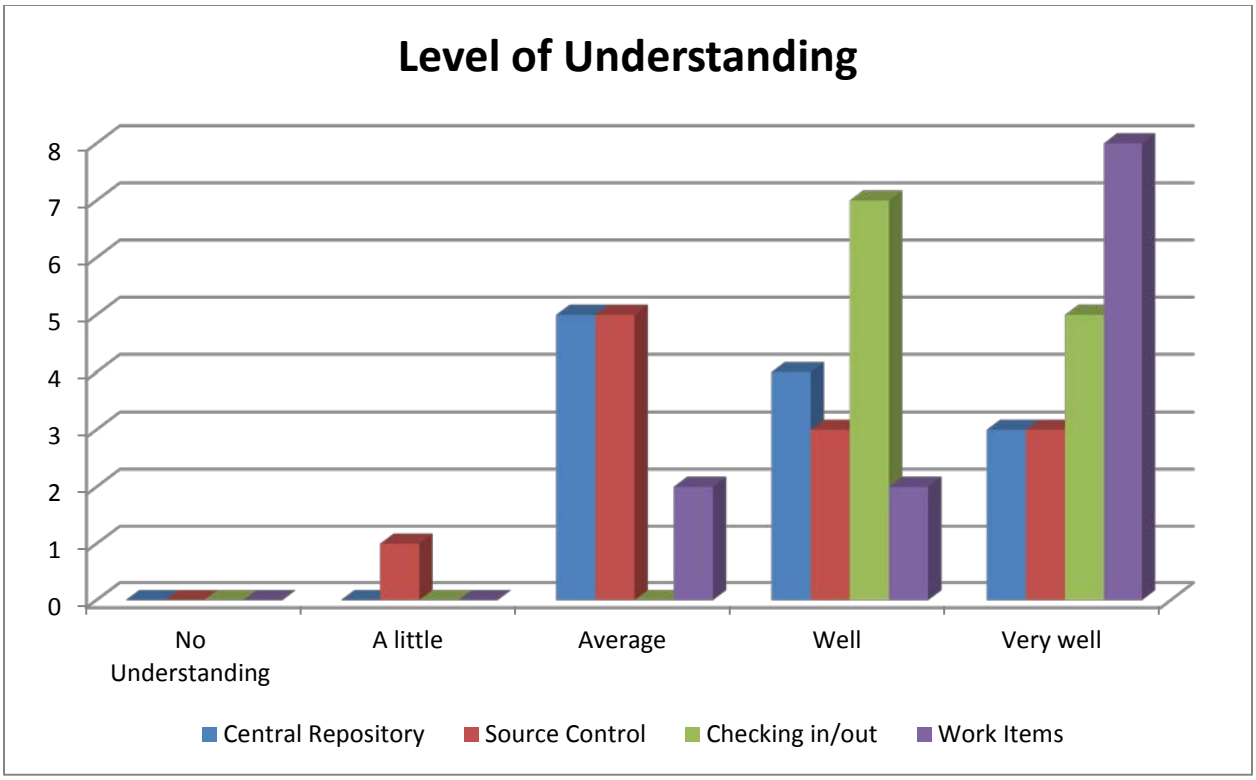
  - Commented that if he went somewhere else to teach, he would not know how to install everything in order to be able to use it.

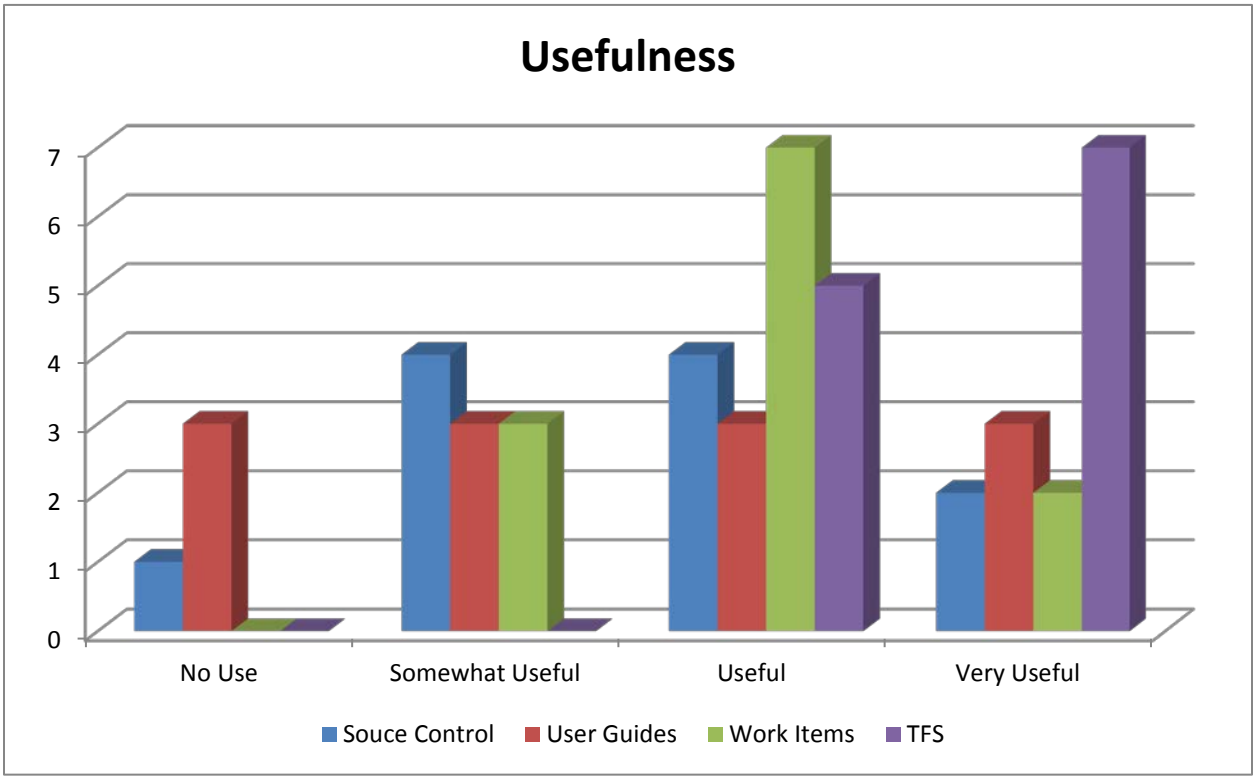
- Did the TFS server need any maintenance?
  - No.
  - The host server did require some hardware maintenance.
- Did anything in TFS need to be modified?
  - No.
- Comments?
  - At the beginning of class, Kevin would discuss what source control is, the benefits, and how it is used in the professional workplace. With the assignments typically being very small and simple, students often did not realize the benefits until the last assignment.
  - For some of the assignments, he gives the students the project template for the assignment. It took some time to push the project to all of the students' workplace. Commented that before, the students would have to download, unzip, do their work, zip, and upload.

Appendix S

Student survey results




Level of Understanding
8
7
6
5
4
3
2
1
0
No Understanding
A little
Average
Well
Very well
Central Repository
Source Control
Checking in/out
Work Items


Usefulness
7
6
5
4
3
2
1
0
No Use
Somewhat Useful
Useful
Very Useful
Souce Control
User Guides
Work Items
TFS