ISSN: 2167-1435
v29 n1988

Teaching Introductory Visual Basic Using Microsoft's Team Foundation Server

Kevin D Matthews
matthewskd@uncw.edu

Paul Martin
pgm0543@uncw.edu

Douglas Kline
klined@uncw.edu

Department of Information Systems and Operations Management
University of North Carolina Wilmington
Wilmington, NC 28403

Abstract

The enterprise-level application lifecycle management system Microsoft Team Foundation Server was used in an introductory Visual Basic programming course for assigning projects, collecting finished projects, and tracking feedback. The sample class implementation is discussed and observations are made. The system was found to add little or no cognitive overhead to the students, saved the instructor and student time on several levels, and introduced professional tools and concepts to the students.

Keywords: introductory programming, version control, team foundation server, teaching, pedagogy

1. INTRODUCTION

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 or low-value tasks. For students, such tasks include: local file management, ad hoc versioning, compressing files, emailing or dropping files on a network shared drive, etc. For instructors, such tasks include: creating an assignment outline, file management, setting up delivery methods, decompressing files, navigation through folder structures, providing written feedback, returning grades, etc. 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.

Ideally, students should be introduced to not only programming concepts, but also professional programming practices. Professional programming involves collaborative software development and other practices that are not well-addressed in many University curriculums. Tools such as version control, task tracking, bug tracking, and workflow management systems are generally not covered in programming courses. If presented, they are rarely used in practice but merely presented as knowledge topics during discussion. As a result, graduates are not well-prepared to use these tools. If employed, they must be trained by their employers; they must adopt professional software development practices after entering the workforce.

While collaborative software development tools and systems were mainly developed to facilitate professional software development teams, they may also be useful in an educational context. They can not only relieve the burden of some non-value-added and low-value tasks, but will also expose students to professional work habits during their educational career.

To test the benefits of introducing professional programming tools, Microsoft's Team Foundation Server (TFS) was used in an introductory Visual Basic .NET programming course in the Department of Information Systems and Operations Management at the University of North Carolina Wilmington. Although the sample implementation focused on VB.NET programming, the concepts apply to programming education in general. Please note that specific details are supplied for TFS which can be utilized by several programming language environments. TFS was coupled with Visual Studios 2010.

2. BACKGROUND AND LITERATURE REVIEW

Motivation

Using TFS, a central repository can be created for all students submitting VB.NET programs. By having a central code source, this eliminates the need for students and instructors to spend time on non-value-added activities including file compression and decompression. This repository eliminates the need for students and instructors to exchange files, gives the instructor access to every student's workspace, allows students to perform all work in one location, and provides file management for the students' files. These advantages aid both the student and the instructor.

One of the areas in programming education that requires additional non-value-added activities is the process of file exchange between the students and instructors. Using TFS to create a central repository for all students' .NET programs allows for easier submission. Instead of requiring that students compress and upload their project files and requiring instructors to decompress files, students will be able to submit their work directly to the central repository. The instructor will have access to all of their students' workspaces at any given time. This feature eliminates the need for any file exchanges.

Submitting, retrieving, and grading files as well as providing feedback are areas in programming education where substantial time must be spent on non-value-added activities. With the use of TFS, these activities can be streamlined. The student will submit work to the central repository and instructors will have direct access to the submitted work. As previously stated, this eliminates the need of file exchanges between the students and instructors. By removing the need of file exchanges, the grading process will be easier on the instructor and save them time. This allows for the possibility of more assignments and multiple iterations of a single assignment. Saved time also allows the instructor to provide better, more substantial feedback to the students. According to current learning theories, these are essential factors in promoting student success.

One of the other huge motivating factors of creating a central TFS server is that it integrates seamlessly with Visual Studio 2010. This is the current Integrated Development Environment (IDE) used in the sample class which was chosen for integration to TFS. If presented properly, students will be able to work in a collaborative development environment using professional tools and practices from the very first class in which a project is introduced. While Visual Studio is a professional tool, it was not previously being used as a collaborative development environment. Integrating Visual Studio with TFS will allow students to use Visual Studio in a collaborative manner which more closely resembles the professional use of the application.

Graduating students are not lacking knowledge in programming; they are merely inexperienced with professional practices and tools. With the use of TFS, students can be exposed to professional practices and tools. The use of TFS with Visual Studio will allow for students to learn the strong work habits that are needed by professional software developers. After recognizing the advantages of integrated tools, students can see the increase in productivity that such tools will allow (Reid & Wilson, 2005).

TFS will also introduce students to a wide variety of professional tools that are currently used in

ISSN: 2167-1435
v29 n1988

the industry. Introducing the concepts of these tools and utilizing them throughout the programming education will improve a student's overall knowledge and better prepare him/her to become professional programmers. Reid & Wilson (2005) also note that seeing instructors use professional tools in the classroom increase a student's likelihood to believe these tools are effective and important.

Learning Theory

According to Robert Gagne (1965, 1985, 1988; Gagne, Briggs & Wager, 1992), nine elements should exist in any lesson in order for a student to learn. These "conditions for learning" form the framework for cognitive learning theory. They are:

- Gaining attention ("reception")
- Informing the learner of the objective ("expectancy")
- Recalling prior learning ("retrieval")
- Presenting the stimulus ("selective perception")
- Providing guidance ("semantic encoding")
- Eliciting performance ("responding")
- Providing feedback ("reinforcement")
- Assessing performance ("retrieval")
- Enhancing retention and concept transfer ("generalization")

Of the nine elements Gagne et al. proposed, other research has shown that eliciting performance ("responding") and providing feedback ("reinforcement") are the conditions most directly associated with student success (Martin, Klein & Sullivan, 2007).

In addition to cognitive learning theory, the models of behavioral learning theory introduce principles of contiguity, repetition, and feedback (Gagne et al., 1992). To uphold these principles, an educator should present multiple similar assignments in a short amount of time. In addition, adequate feedback that not only identifies correct answers but also provides an explanation and rationale for incorrect answers is required (Debuse, Lawley & Shibl, 2007).

Finally, in 1996, Rakes first recommended increasing student success by shifting from traditional learning theories to a resource-based view of learning. This view places the instructor as a guide who provides resources instead of the traditional expert who mainly dispenses knowledge (Rakes, 1996).

The previous overview of learning theories and concepts shows that learning is promoted when several key aspects of a student's classroom experience are present. The following concepts were utilized when developing the implementation for TFS within the classroom:

- A response from a student should closely follow any instructions.
- Students should be presented with several closely related assignments in order to reinforce concepts.
- Feedback should be immediate and customized to identify correct answer and errors that are present.

3. CLASS IMPLEMENTATION

Specifically, this sample implementation involved:

- Creating workflows for student project assignment, submission, feedback, grading, etc.
- Setting up a TFS system for the Introduction to Visual Basic programming class

Team Foundation Server Workflow

The basic workflow for programming assignments includes:

- Instructor assigns projects as a Work Items within TFS
- Student accepts Work Item from TFS
- Student solves problem by coding solution files with Visual Studio
- Student performs a check-in of solution files within Source Control
- Student completes Work Item and assigns to the instructor for grading
- Instructor accepts Work Item from TFS
- Instructor provides feedback in source code or in Work Item details and assigns to the student as graded or incomplete (needs work)
- Student views grade or makes corrections

Note that the entire workflow above is version controlled and tracked within TFS. Every change and every step in the workflow is recorded, including authorship of changes. Also note that the workflow does NOT include any file management (non-value-added) activities as Visual Studio handles these tasks through the connection to TFS. The basic graphical workflow as seen in Visual Studio is shown in Appendix A.

Once TFS has been set up, the "day to day" actions for both the instructor and student took place inside Visual Studio.

It should also be noted that the implementation had two main prerequisites:

1. Student and Instructor user accounts already existed on a Microsoft Exchange Server. This server was located on the same domain as the new TFS server.
2. The Instructor who initially set up the course had the proper admin permissions within TFS.

Without these requirements being met, the setup and implementation would not have been possible.

Security and Organization

With TFS set up and Visual Studios running, the instructor created a new Team Project for the class. The Team Project was given a name. For the sample implementation, the project name was SP12MIS216 to designate the semester and course.

After creating a Team Project, Security settings were created. To utilize the workflow from this implementation, two new groups of users were created: Students and Instructors. Each group was given specific permissions within the Team Project. Membership for all students and instructors for the Team Project were then created for the newly established groups. Student exchange accounts were added to the Students group and instructor exchange accounts were added to the Instructors group. This provided the initial groundwork for security permissions for this specific Team Project.

Another major part of TFS that easily allows the transition into the academic setting is the existence of areas and iterations. Areas are defined by Microsoft as "an organizational hierarchy of components and features" (Microsoft, 2012). Iterations are defined as "a process hierarchy of events in the project life cycle" (Microsoft, 2012). For the class implementation, an area was created for each student and an iteration was created for each assignment. Student email ids were used as the area names while the actual assignment titles were used for the iteration names. This setup allowed the creation of the desired tree structure for a class environment. All students had their own branch (an area), and each branch had the same sub-branches (iterations). Once set up, each student was granted specific privileges to their respective area.

The final security measure involved setting up Source Control with permissions under the Team Project. Every student was given his/her own folder. These folders were created within the Team Project "root" folder within Source Control. Student email ids were used for the folder names. Once created, each student was given permissions within his/her folder only.

For specific security settings listed above, see Appendix B.

Benefits to the Instructor

For the instructor, an add-on known as TFS Power Tools was installed. This add-on is available for free in download form from Microsoft. The add-on provides the tools such as the Process Editor and the creation of Work Item Templates. Templates allow easy replication of Project Tasks that need only be created once. This feature greatly decreased some of the repetitive tasks involved with initial setup of assignments.

With Power Tools installed, the instructor created a Work Item Template for Tasks within the Team Project. The template included the details for one assignment; Project Tasks each represented one assignment. When the assignment details were finalized and ready to be distributed to students, the template was activated to create new Work Items with the given details. One Work Item was created for each student in the class. Since the template had all assignment details, the instructor only had to modify the Area and Assigned To fields for each Work Item. The area was the student's email id while the Assigned To field was the student's exchange account from the Students group. With the workflow shown in Appendix A, all Tasks start in the Student Work state.

Each time a student completed and checked-in files under Source Control, the instructor had access to the code. The instructor opened Source Control and performed a Get for the latest versions. This was either done on a Project "root" folder level or on individual student folders. Once the Get was initiated, all files were downloaded and the organization set forth in the aforementioned Team Project design was maintained. All student files were then accessible within Source Control with one instance of Visual Studio.

Not only were all files accessible through Source Control, but all aspects of the project are available. The instructor was able to view previous versions of the code, check-in times, and many other details. This allowed the instructor to see how the student's code had progressed throughout the assignment's lifetime.

Since each assignment was given as a specific Project Task, the details of that Task were available to the students. Feedback was able to be quickly and easily supplied in one or both of two forms: comments within the source code or comments within the Task details. These methods were both available through the current connection to the Team Project in Visual Studio.

Another advantageous feature within each Team Project was the availability of queries. The instructor was able to create custom queries that were shared among all students. Such shared student queries showed assignments in progress, assignments awaiting grades, graded assignments, etc. In addition, hidden queries were created. Hidden queries for instructors listed details for outstanding assignments, assignments awaiting grades, and grade reports.

Benefits to the Student

Students shared the same file structure, code versioning, Task detail availability, and query possibilities as instructors. The central location of all files and assignment details added the same time benefits for students. This also reduced the complexity of having to keep track of current versions of projects scattered across local drives, network share drives, and portable jump drives. With the Source Control features of TFS, students no longer needed to upload or e-mail their work to the instructor for grading or review; all aspects of programming assignments were gathered inside Visual Studio.

In addition to time savings, the experience with the new tools and features of Visual Studio and TFS were a great benefit to students. Basic programming skills have and will continue to be introduced in the introductory programming course. The introduction to professional tools simultaneously with programming concepts allowed students to become competitive in a professional programming environment. By arming students with these essential skills, they earn a more robust education and obtain the necessary skills to perform from the beginning.

Many introductory programming courses are targeted towards students with no programming experience. When they learn the professional tools simultaneously with coding principles, the learning curve of these new tools can be coupled with the learning curve of programming. This prevents students from having to "re-learn" their process of programming once it has been established. The reduction in cognitive overhead is a benefit to introducing these professional tools and concepts as they are starting to learn programming skills.

4. LESSONS LEARNED

The use of Team Foundation Server can provide several advantages to both instructors and students in an introductory programming course. These benefits are recapped below. In addition, some issues that were noted during the sample implementation are reviewed and possible solutions are given.

Time Savings

Through the central repository of data and files created with TFS, instructors and students save time throughout the course of a single class. These time savings range from file management time to feedback and grade management time. All files are located within the Visual Studio Source Control interface. This eliminates any file compression, submission, decompression, or organization by all parties involved. In addition, details of each assignment are included in the TFS Work Item within Visual Studio's Team Foundation panel. These details include instructor expectations, student questions or comments, and any associated feedback.

With the sample implementation, it was estimated that the time savings for each assignment was equal to approximately one hour for the instructor. With ten projects within one class, this would equate to ten hours in time savings per class. The saved time was devoted to value-added benefits such as providing feedback, lesson modification and enhancements, and one-on-one time with students.

The estimated time savings for each assignment was approximately thirty minutes for students.

In our sample implementation, approximately five hours of time was saved per student. This time was used to interact more with the instructor, write more code, and implement changes based on feedback.

Tie to Learning Theory

Prior to the sample implementation, a file compression and upload method was used to receive files from students. Because of the involved nature of this process, some key elements from lessons were impacted. To save time, instructors were reducing the number of assignments, delaying feedback, and limiting the amount of feedback per assignment.

Due to increasing class sizes, a limited number of instructors, and the time investment created by the previously noted non-value-added tasks, instructors were not able to produce the quality and quantity of feedback needed to promote student success. In addition, feedback should be received immediately in order to learn (Gagne et. al., 1992). Prior to the addition of TFS, grading times would take up to two weeks depending on the complexity of the assignment. With the use of TFS in the sample implementation class, grading times were reduced to a maximum of four days. Due to the ease of retrieving and accessing files, the instructor was striving to provide feedback before proceeding to another assignment or project. This allowed students to immediately learn and implement modifications based on the feedback they received.

Exposing Students to Professional Practices

Learning to program is just the first step in a student's education. To become an effective professional programmer, they must also learn how to apply this knowledge with professional practices and tools. Prior to the use of TFS in the classroom, Visual Studio was being used just as an IDE. Unfortunately, other professional practices were not applied and many available tools were not utilized. By introducing TFS, students were introduced to a professional collaborative development environment including version control. All work on projects followed professional practices using professional tools. This approach produced more well-rounded students who obtained first-hand experience with some of the necessary skills and knowledge to become professional programmers.

Up Front Time-Invested

As detailed further in Section 3, there does exist a certain level of initial set up for each class. This involves setting up the Team Project, creating the organization required to utilize TFS in a classroom setting, and utilizing security settings as needed. It was found that these settings could be performed in approximately two hours. This time-investment is seen as equivalent to the time involved in setting up the network share that was used prior to the use of TFS.

Security Issues

While many security permissions were set to allow students access and to restrict features as required, there were some security issues that should be noted. Perhaps the most prevalent was that seen by the query feature within TFS. While this is a great benefit to view progress and disperse quality data to students and instructors, it is possible for students to modify queries and view data that they should not have access to. The main concerns were other student's grades.

As a temporary resolution in the sample implementation, grade values were kept in a third party course management software package. Feedback was provided within TFS for each assignment. While this does introduce the overhead of utilizing two different systems, the grade protection benefits were far greater. While students did not have the ability to modify the data, there needs to be further research into restrictions that would eliminate students from viewing this data.

5. CONCLUSIONS

By introducing Team Foundation Server into an already existing introductory programming course, instructors and students can utilize a central repository of code and the TFS features not typically introduced in these courses. Benefits include:

- Time savings
- Performing many tasks within one application (Visual Studio)
- Adding experiences with professional tools and practices
- More rapid feedback distribution
- A decrease in non-value-added tasks

6. REFERENCES

Debuse, J., Lawley, M., & Shibl, R. (2007). The implementation of an automated assessment feedback and quality assurance system for ICT courses. *Journal of Information Systems Education*, 18 (4), 491-502.

Gagne, R. (1965). The Conditions of Learning (1st Ed.). New York: Holt, Rinehart & Winston.

Gagne, R. (1985). The Conditions of Learning (4th Ed.). New York: Holt, Rinehart & Winston.

Gagne, R. (1988). Mastery Learning and Instructional Design. *Performance Improvement Quarterly*, 1(1), 7-18.

Gagne, R., Briggs, L. & Wager, W. (1992). Principles of Instructional Design (4th Ed.). Fort Worth, TX: HBJ College Publishers.

Martin, F., Klein, J. & Sullivan, H. (2007). The Impact of Instructional Elements in Computer-Based Instruction. *British Journal of Educational Technology*, 38(4), 623-636.

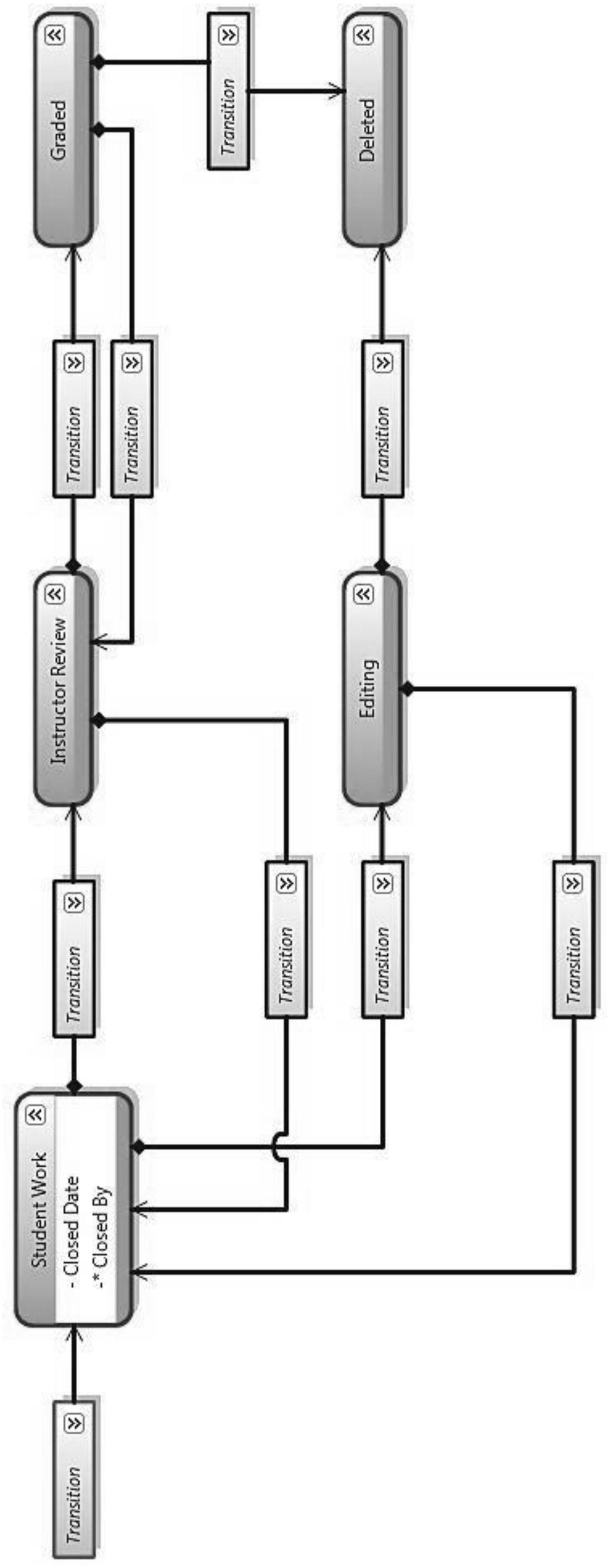
Microsoft (2012). *Setting Initial Project Areas or Iterations*. 15 August 2012. Available at http://msdn.microsoft.com/en-us/library/ms181497%28v=vs.80%29.aspx.

Rakes, G. (1996). Using the Internet as a tool in resource based learning environment. *Educational Technology*, 6(2), 52-29.

Reid, K., & Wilson, G. (2005). Learning by Doing: Introducing Version Control as a Way to Manage Student Assignments. *SIGCSE Bulletin*, 37(1), 272-276.

ISSN: 2167-1435
v29 n1988

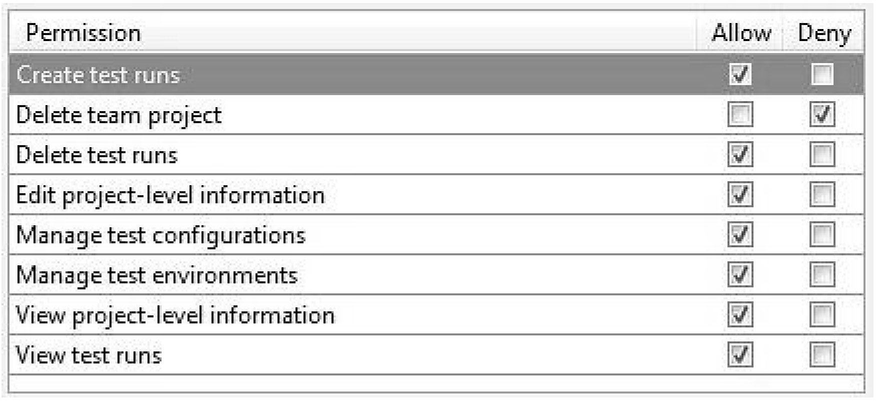
Appendix A: Workflow Utilized for a Team Project Task Work Item



ISSN: 2167-1435
v29 n1988

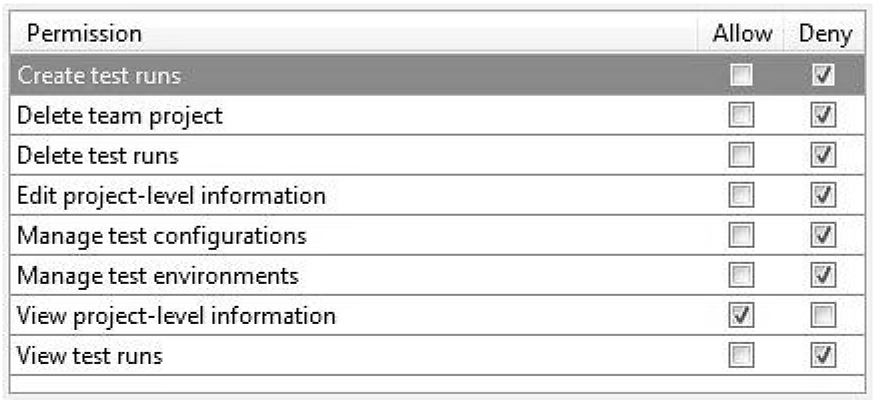
Appendix B: Specific Permission Settings

Permissions for Instructors Group



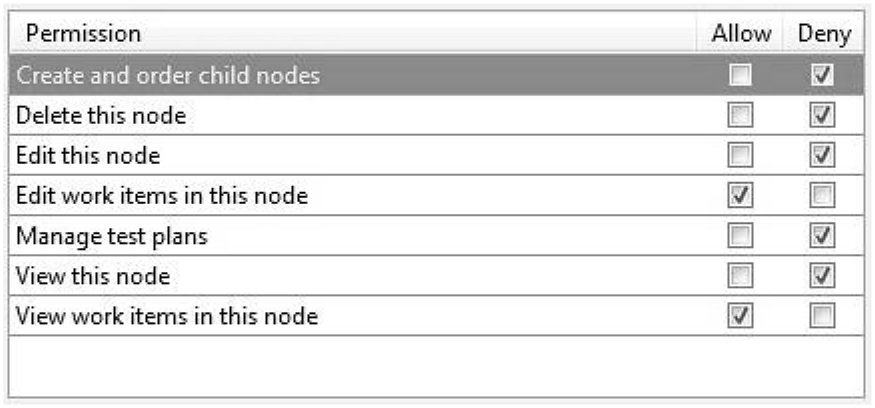
Permission	Allow	Deny
Create test runs	✓	
Delete team project		✓
Delete test runs	✓	
Edit project-level information	✓	
Manage test configurations	✓	
Manage test environments	✓	
View project-level information	✓	
View test runs	✓	

Permissions for Students Group



Permission	Allow	Deny
Create test runs		✓
Delete team project		✓
Delete test runs		✓
Edit project-level information		✓
Manage test configurations		✓
Manage test environments		✓
View project-level information	✓	
View test runs		✓

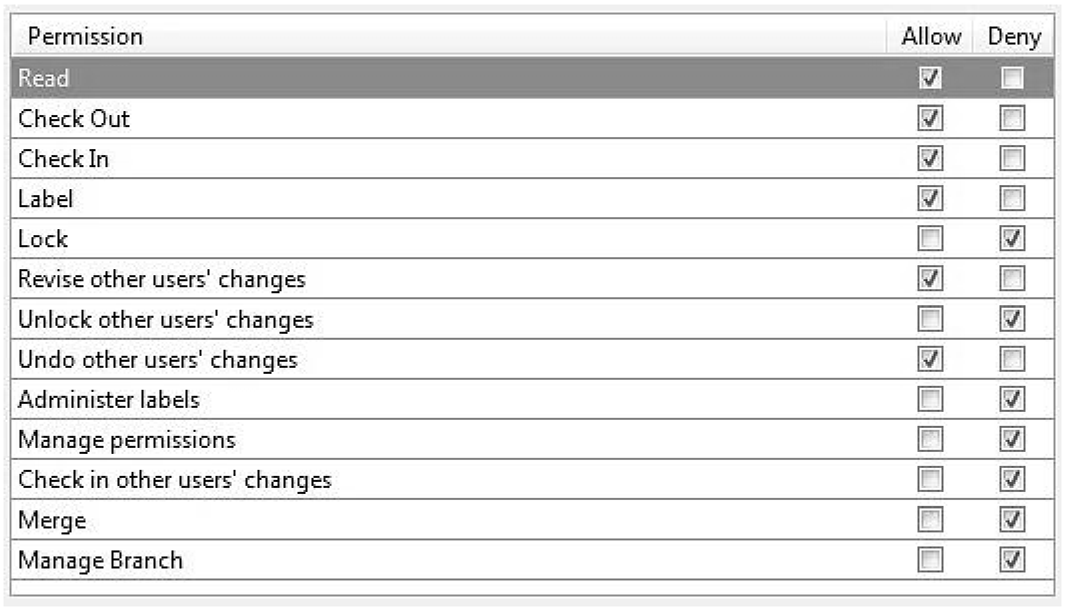
Permissions for Student in Area



Permission	Allow	Deny
Create and order child nodes		✓
Delete this node		✓
Edit this node		✓
Edit work items in this node	✓	
Manage test plans		✓
View this node		✓
View work items in this node	✓	

ISSN: 2167-1435
v29 n1988

Permissions for Source Control Folder



Permission	Allow	Deny
Read	✓	
Check Out	✓	
Check In	✓	
Label	✓	
Lock		✓
Revise other users' changes	✓	
Unlock other users' changes		✓
Undo other users' changes	✓	
Administer labels		✓
Manage permissions		✓
Check in other users' changes		✓
Merge		✓
Manage Branch		✓