Практикум по курсу «Введение в программную инженерию» на базе Microsoft Visual Studio Team System

Оглавление

0. О практикуме

0.1 Общее

Практикум предназначен для практического усвоения ряда положений программной инженерии и использует в качестве инструментария продукт Microsoft Visual Studio Team System, поддерживающий многие практики командной разработки ПО. В рамках практикума предполагается освоение планирования, конфигурационного управления (средства контроля версий и управление сборками), автоматического тестирования и командной работы в рамках формально определенного процесса разработки. Такой выбор обуславливается с, одной стороны, важностью этих практик в реальном промышленном производстве, с другой стороны, возможностями, предоставляемыми продуктом MS VSTS. Ряд важных аспектов – проектирование, разработка требований и т.д. не вошли в данный практикум по причине ограниченности времени – он рассчитывается на один университетский семестр. Мы старались предложить максимально эффективный способ обучения основам программной инженерии в рамках университетского процесса обучения, прекрасно понимая, что полное освоение практик включенных в данный практикум, а также и иных, равно как и продукта MS VSTS, возможно при погружении в реальный индустриальный процесс.

Особо отметим, что нашей задачей является не столько обучение продукту MS VSTS, сколько использование его в качестве основы для практического освоения программной инженерии. В современном производстве без подобных инструментов уже не работают, и в принципе, для обучения годится любой продукт такого класса. Достоинство продукта MS VSTS заключается в его доступности для целей обучения. Компания Microsoft ведет широкую работу по бесплатному распространению своих продуктов в образовательных целях в российских университетах, поэтому и данный продукт не составляет труда получить. Однако для его включения в учебный процесс от преподавателей требуется индустриальный опыт использования среды разработки Microsoft Visual Studio – важной составной части MS VSTS – и существенная поддержка в области администрирования (либо собственный опыт, либо поддержка опытного сетевого администратора). При этих условиях ознакомление с TFS (серверной компонентой VSTS) и рядом удобных клиентских приложений (инструменты пакетов Visual Studio Team Suite и Team Foundation Server Power Tools) не составит существенного труда, но потребует значительной работы. Мы надеемся, что данные методические материалы помогут ее сократить и облегчить.

0.2 Требований к техническому оснащению

Занятие по данному курсу должны проводиться в компьютерных классах, удовлетворяющих следующим условиям.

- Все компьютеры должны находиться в домене Active Directory под управлением доменного контроллера версии 2003.
- Все учащиеся должны иметь логин и пароль для входа в этот домен.

- В рамках этого домена должен быть развернут Team Foundation Server 2003 со всеми необходимыми компонентами (Microsoft SQL 2005, Sharepoint Server 3.0, Internet Information Server 6,0[1]).

- В рамках этого домена должен быть развернут Team Foundation Build и настроен для работы с Team Foundation Server.

- На всех компьютерах класса должна быть установлена Visual Studio Team Suite 2008 и Team Foundation Power Tools[2].

- На сервере TFS должен быть импортирован шаблон процесса разработки Scrum for TFS[3].

Каждый участник практикума должен иметь по компьютеру. Также целесообразно, чтобы в классе была доска, проектор. По ходу практикума могут всплывать различные общие вопросы, пробелы и пр. – и тогда на некоторое время занятия превращаются в лекции.

0.3 Организация процесса

Занятия предполагается проводить со студентами, разбитыми на группы в 5-6 человек. Общее количество одновременно обучающихся команд может составлять 2-4 в одном классе. Возможен также запуск нескольких параллельных команд в разных классах, по разному расписанию и пр. Более крупные группы нежелательны, поскольку существуют разовые, общие для всей группы действия – создание проекта, изменение параметров процесса и пр. – и они потеряются в более крупной группе.

В качестве методологии процесса разработки ПО мы предлагаем использовать Scrum. Эта методология очень проста и хорошо проецируется на учебные команды. Кроме того, Scrum получил в последнее время значительное распространение в мире и в России. Наконец, имеется и легко доступен Scum-шаблон для MS VSTS.

Теперь о проекциях Scum на наш учебный процесс. Product Owner – это преподаватель, в роли Scrum-мастера может выступать один из студентов – самый активный, самый заинтересованный. При этом данная роль может допускать широкие вариации по активностям и ответственностям – от простого слежения за временем и некоторых формальностей (именно Scrum-мастер будет выполнять некоторые общие, единые для всех действия, а остальные будут наблюдать за тем, как он это делает), до некоторых функций project manager. Здесь многое зависит от самих студентов – найдется ли в группе один, которому предмет будет интереснее всех, кто будет самым активным. Часто такие студенты находятся, но иногда и нет. В последнем случае большинство обязанностей Scum-мастера возьмет на себя преподаватель. Ну, и наконец, вся группа учащихся будет являться scrum-командой, работающей над реализацией некоторого проекта в рамках одного sprint. Начальные требования к модельной задаче предоставляются в виде backlog, из которого студенты изготовляют sprint backlog.

---

[1] При использовании с Windows Server 2008 необходимо использовать IIS 7.0.

[2] http://msdn.microsoft.com/en-us/tfs2008/bb980963.aspx.

[3] http://scrumforteamsystem.com/en/default.aspx.

Исходя из нашего опыта предпочтительно, чтобы практикум курировало два преподавателя – один более осведомленный в индустриальных реалиях (в том числе, прямо из индустрии), другой в большей степени университетский педагог.

0.4 Модельная задача

Данный практикум проводится на основе некоторой практической задачи, которую scrum-команда учащихся реализует в рамках практикума. Требования к этой задаче следующие.

- Она не должна быть очень трудной и допускать реализацию студентами за 6-8 часов.
- Задача должна быть распределенная, разные ее части, розданные разным участникам, должны быть зависимы друг от друга.
- Задача должна иметь более широкий контекст, чтобы ее можно было выделить из более объемлющего backlog.
- Задача должна хорошо подходить для написания модульных тестов.
- Задача НЕ должна содержать сложного пользовательского интерфейса или других технических сложностей.

В качестве своего варианта мы предлагаем предлагаем реализацию игры «балда» на компьютере. Правила игры можно найти здесь: http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B4%D0%B0_(%D0%B8%D0%B3%D1%80%D0%B0).

Команде необходимо сформулировать задачу таким образом, что нужно реализовать не столько отдельное ГУИ приложение с игрой, сколько библиотеку классов, позволяющую легко реализовывать различные вариации этой игры с разным пользовательским интерфейсом или без оного. Кроме того, одной из задач, стоящих перед командой, должна быть реализация компьютерного алгоритма этой игры.

Перед проведением курса преподавателю необходимо составить список пользовательских историй, описывающих необходимую функциональность задачи. Описание должно быть достаточно детальным, чтобы максимально точно определить структуру будущей системы и направить работу студентов в нужное русло.

0.5 Требования к студентам

Для полноценного участия в практикуме студенты должны владеть следующей информацией и практическими навыками.

- Они должны быть знакомы с курсом «Введение в программную инженерию»
- Они должны владеть практическими навыками работы с языком C# и средой разработки MS Visual Studio

0.6 О масштабируемости практикума

Практикум состоит из пяти занятий. Но это не означает, что это ровно 5 встреч преподавателя со студентами. Одно занятие может выполняться на нескольких встречах. Кроме того, в рамках нескольких встреч должна происходить разработка модельной задачи. В зависимости от подготовленности студентов может также потребоваться дополнительная информация, дополнительные детали по тем или иным основам MS VSTS, в частности по модульному тестированию. Этот целесообразно оформлять как отдельную лекцию. В целом мы не старались создать полный guideline оставляя значительную свободу для импровизаций, имея также в виду различную подготовленность и активность студентов и, кстати, преподавателей.

0.7 Обзор тем и задач

Тема 1. При изучении первой темы (на первой серии занятий) студенты организуются как Scrum-команда. Они также знакомятся с условиями модельной задачи, настраивают инфраструктуру TFS для будущей разработки (создают командный проект и распределяют права на работу с ним).

Тема 2. На второй серии занятий студенты практикуются в планировании работ на основе методологии Scrum, а также изучают способы использования системы отслеживания задач TFS. Студентам необходимо импортировать список пользовательских историй их файла Excel в TFS, а затем детально спланировать будущий спринт и распределить задачи.

Тема 3. Третья серия занятий предполагает основную работу по реализации решения модельной задачи. На занятиях этой серии студенты должны также освоится с системой контроля версий TFS, её интеграцией с системой отслеживания задач, а также попрактиковаться в создании ветвей и интеграции изменений.

Тема 4. На занятиях четвертой серии студенты практикуются в разработки модульных тестов средствами Visual Studio Team Developer. На этих занятиях студенты должны освоить средства автоматической генерации тестов, наполнить сгенерированные тесты содержимым, а также научится изменять конфигурацию запуска модульных тестов и считать тестовое покрытие.

Тема 5. Пятая серия занятий посвящена системе автоматических сборок TFS. На этих занятиях студенты должны создать несколько определений для автоматической сборки (build definitions) в разных случаях – с тестами и без, с анализом кода и без и т.д. Кроме того, студенты должны настроить параметры непрерывной интеграции и рассылки уведомлений.

Тема 6. На заключительной, шестой, серии занятий студенты должны повести ретроспективный анализ выполненного Scrum sprint, выявить потенциальные способы оптимизации, а затем и применить их, используя средства настройки процесса разработки TFS. На этих занятиях студенты должны освоить изменение настроек системы остлеживания задач средствами Team Foundation Server Power Tools.

Тема 1. Знакомство и создание проекта.

Целями данного занятия является следующее.

1. Разделить студентов на scrum-команды, определить и обсудить Scrum-роли.
2. Провести начальное знакомство с модельной задачей, которую предстоит реализовать.
3. Прояснить открытые вопросы по модельной задаче с Prodcut Owner.
4. Создать командный проект на TFS и добавить в него пользователей.

Для разделения студентов на scrum-команды и выделения Scrum-мастеров можно прибегнуть к жеребьевке, после чего перейти к знакомству с задачей. Для этого руководитель курса должен заранее приготовить следующие документы:

- Краткое описание основной концепции разрабатываемого приложения.
- Список задач в формате product backlog.

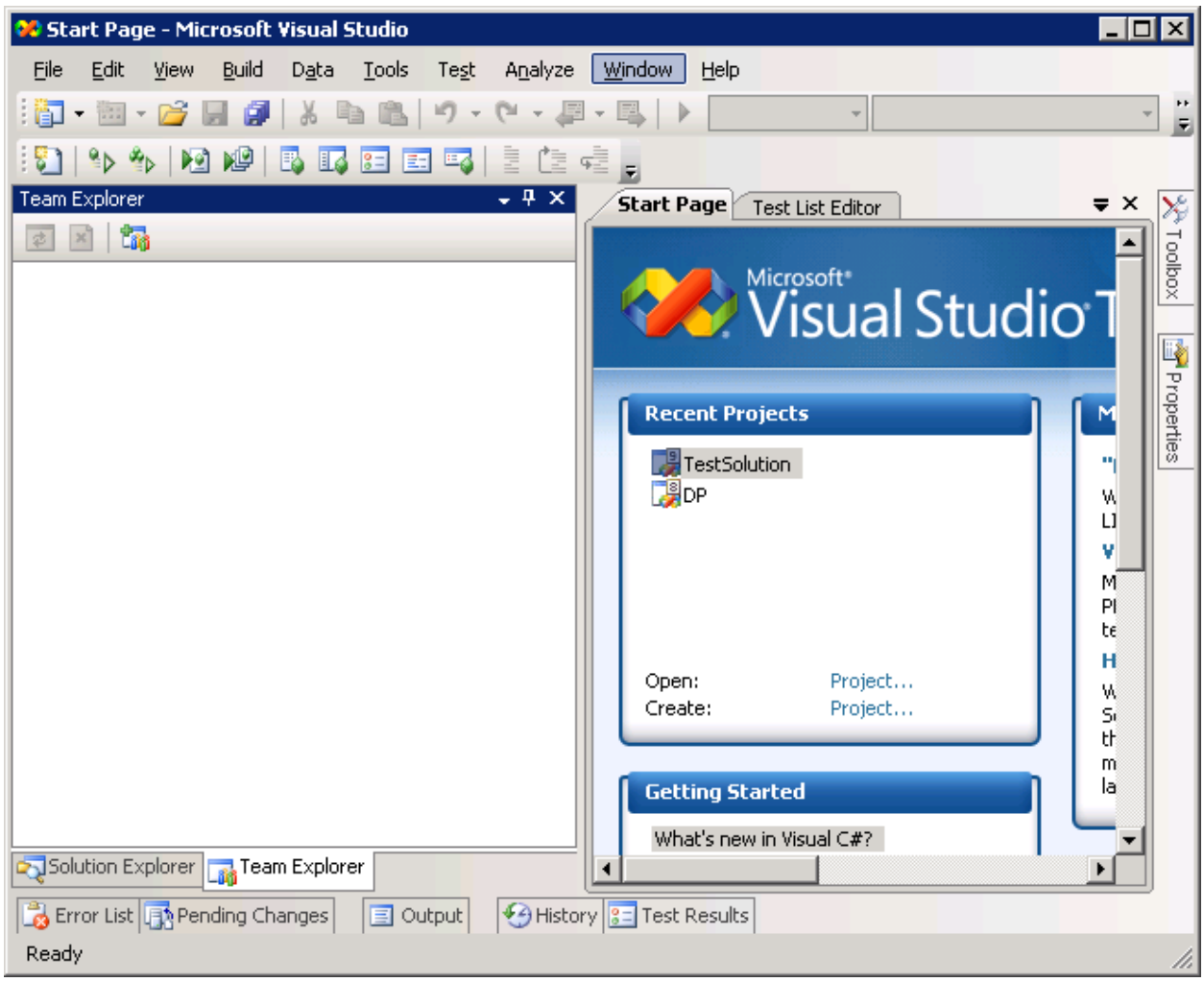
Команды получают эти документы и в течение 20-30 минут обсуждают их в рамках команды, готовя вопросы к хозяину продукта. Затем, в течение 20-30 минут хозяин продукта отвечает на подготовленные командами вопросы.

После того, как команды получили представление о разрабатываемом продукте (модельной задаче), им необходимо создать командный проект в MS VSTS и занести всех участников проекта в список пользователей.

Шаг 1. Создание проекта.

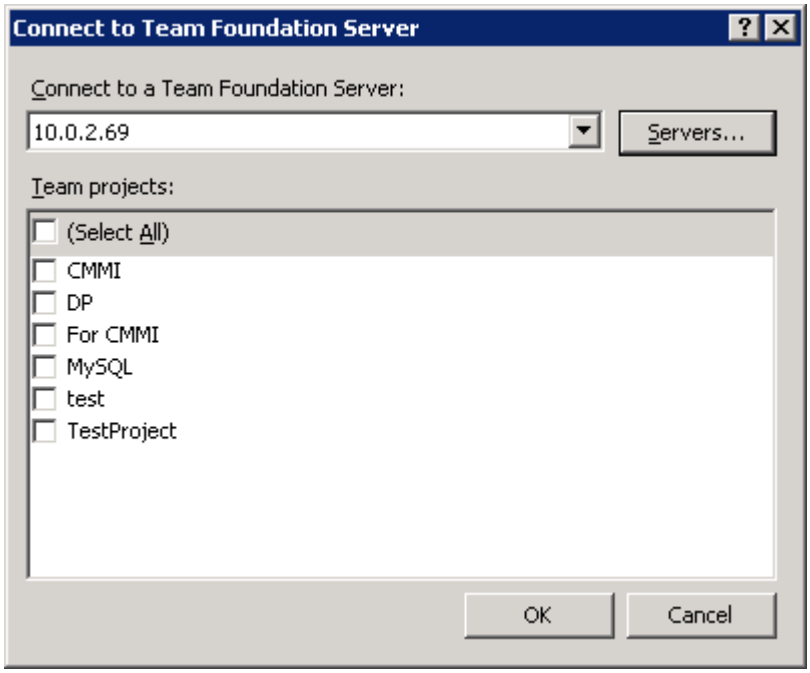
Создание командного проекта осуществляется лидером команды по следующему сценарию:

1. Открыть Visual Studio Team Suite и окно Team Explorer в ней:

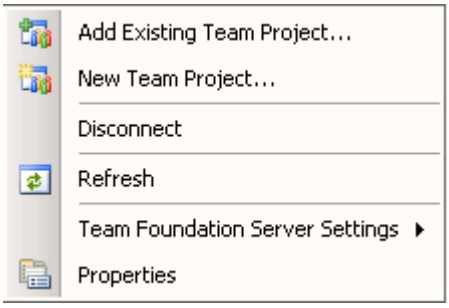


2. Нажать кнопку соединится с сервером  .

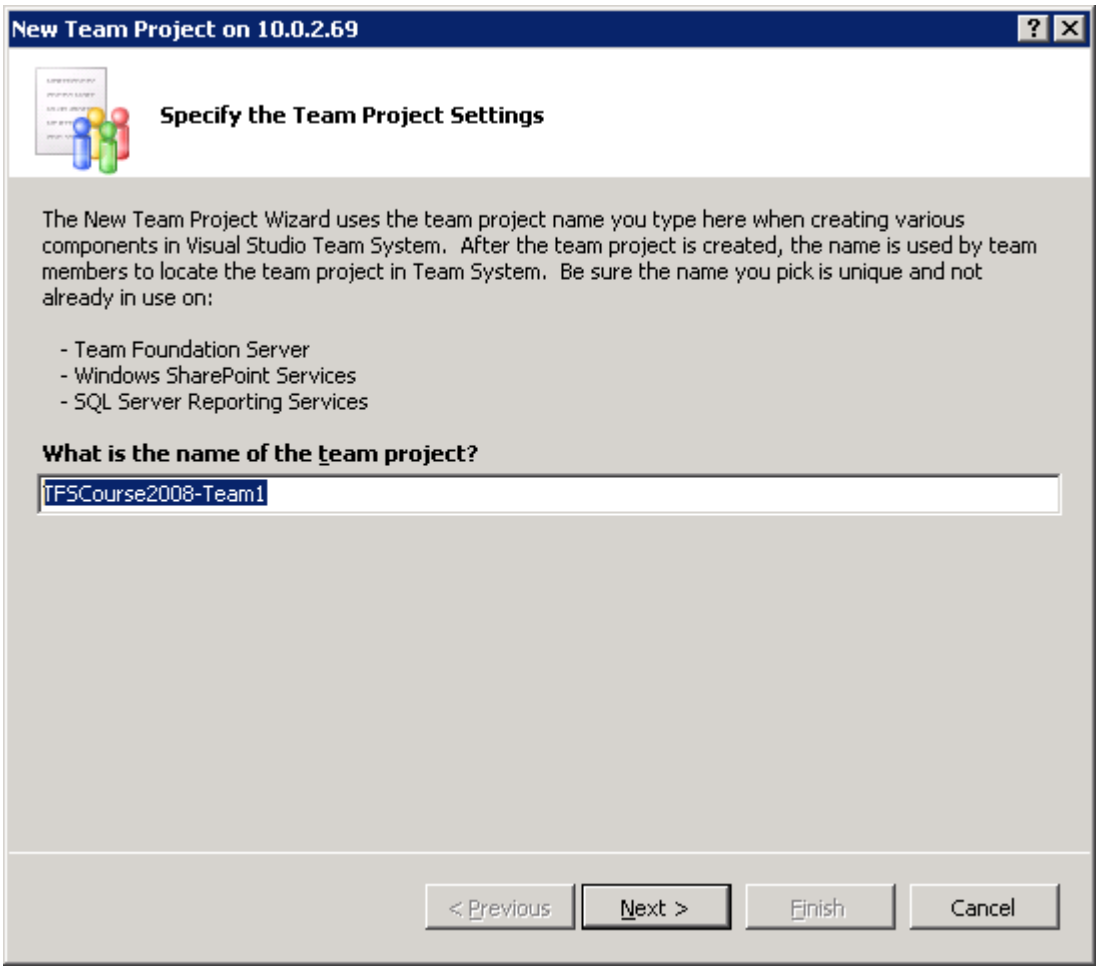
3. Задать имя и порт сервера в открывшееся диалоге:



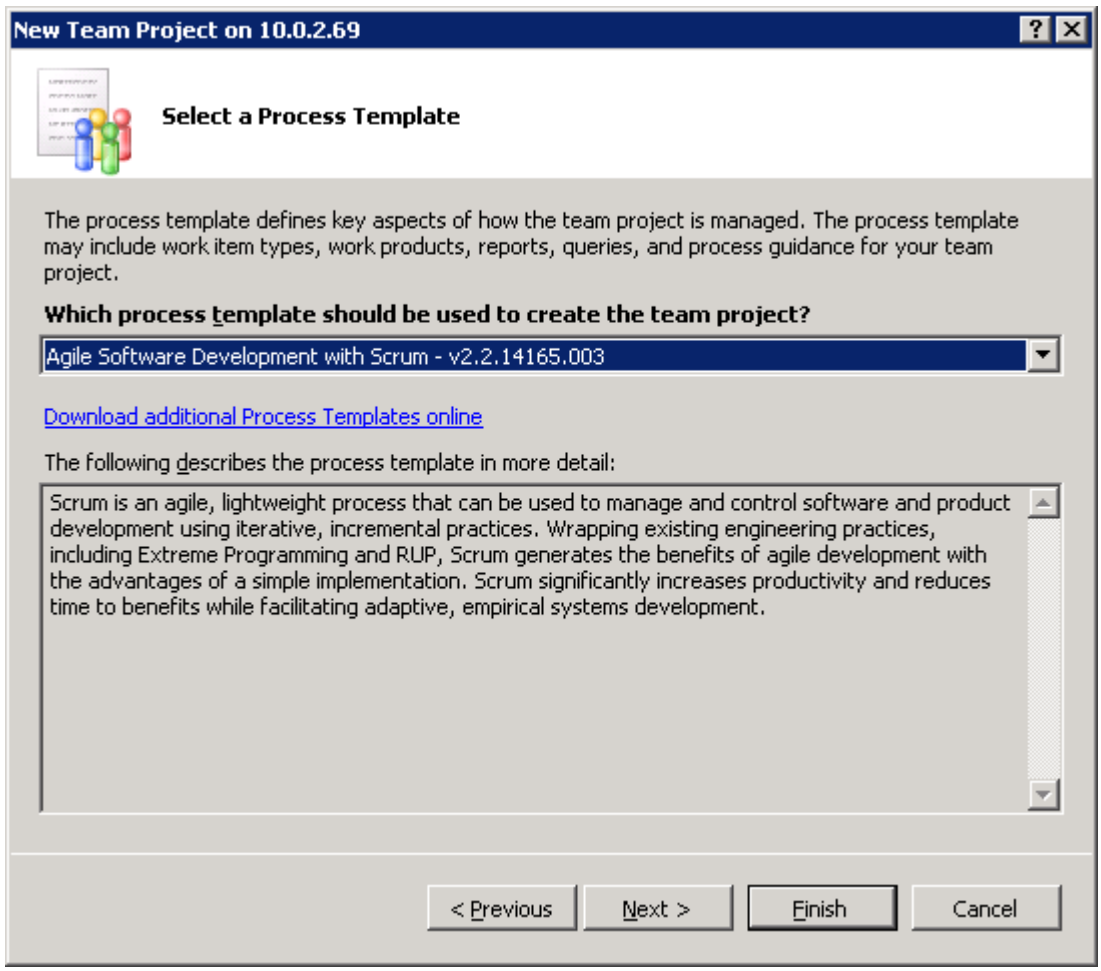
4. После того, как соединение с сервером установлено, запустить процедуру создания проекта, используя соответствующую команду контекстного меню (New Team Project…:



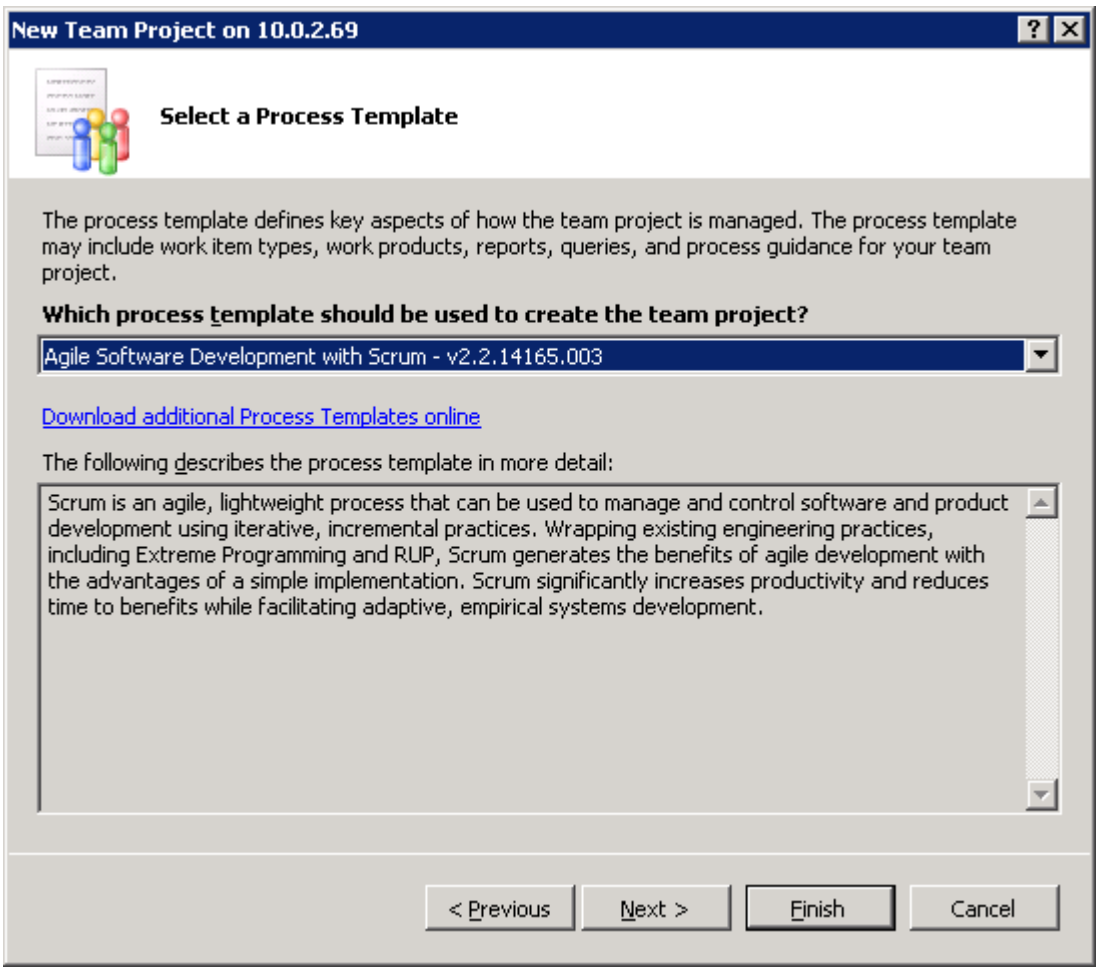
5. В качестве имени проекта необходимо указать TFSCourse<год>-<имя команды>:



6. В качестве шаблона процесса разработки выбрать Scrum:



7. Создать новый пустой раздел в системе контроля версий для данного проекта:

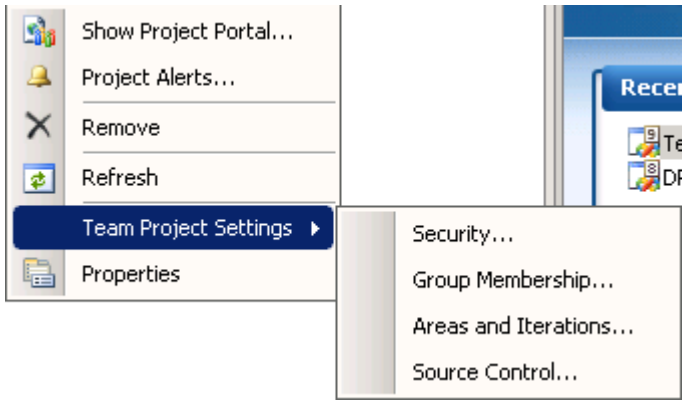


8. После выбора всех настроек, создать проект.

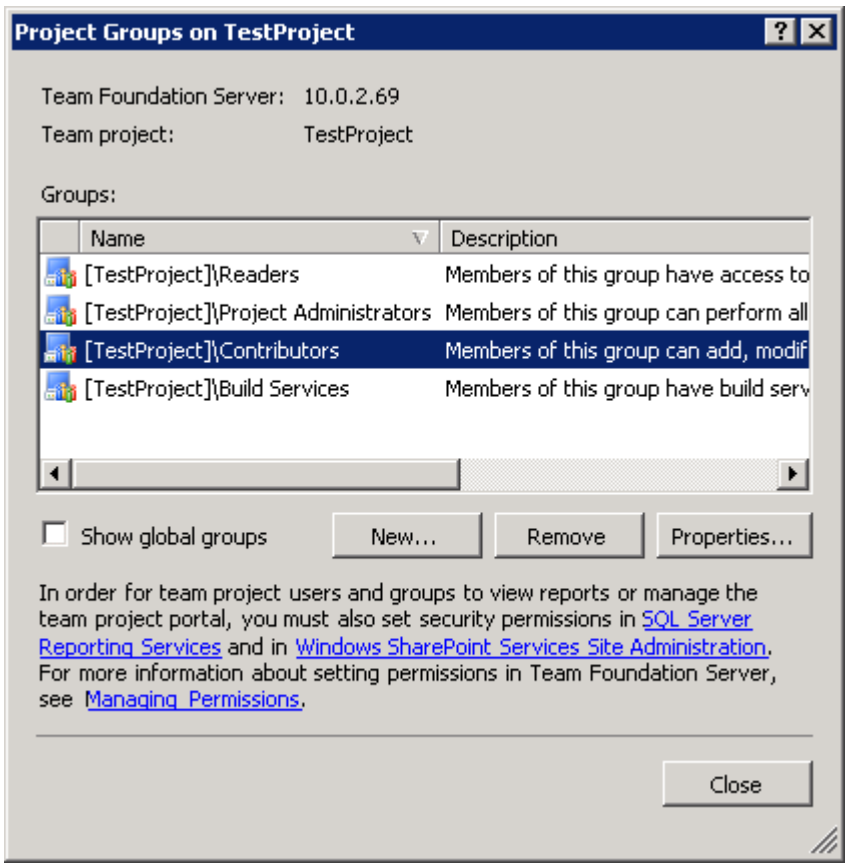
Шаг 2. Настройка прав.

После того, как проект был создан лидеру необходимо выделить права остальным участникам команды для работы с этим проектом. Для этого ему нужно:

1. Выбрать в свойствах проекта раздел Group membership:



2. В открывшемся диалоге включить всех участников в группы Readers и Contributors:



Шаг 3. Подключение проекта остальными участниками команды.

После того, как все получили права на работу с проектом, каждый участник команды должен на своей машине открыть Visual Studio и добавить соединение с этим проектом. Выполняется это аналогично пунктам 1-4 первого шага занятия.

Тема 2. Работа с системой отслеживания ошибок.

Основной целью данного занятия является знакомство участников с системой отслеживания элементов работы.

1. Создание элементов работы средствами Visual Studio и Team Explorer.
2. Импорт и экспорт элементов работы из/в Mcrosoft Excel.
3. Назначение ответственных за элементы работы.
4. Отслеживание текущего статуса посредством отчетов.

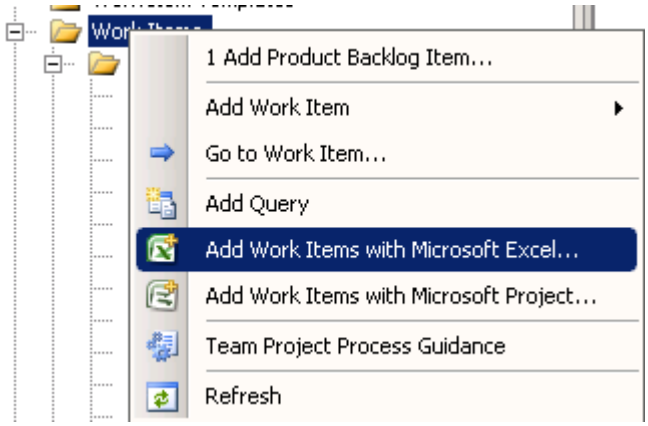
В рамках данного занятия предполагается провести планирование работы команды по методологии Scrum. Команды уже провели предварительное знакомство с проектов, а на данном этапе от них требуется следующее.

1. Импортировать содержимое списка требований в TFS, используя средства импорта из Excel.
2. Подробно рассмотреть 10 наиболее приоритетных пользовательских историй.
3. Обсудить возникшие вопросы с хозяином продукта.
4. Провести детальное планирование и разбить пользовательские истории на мелкие подзадачи.
5. Распределить подзадачи среди участников проекта.
6. Отчитаться перед хозяином продукта о том, какие пользовательские истории были запланированы. При отчете использовать отчеты TFS.

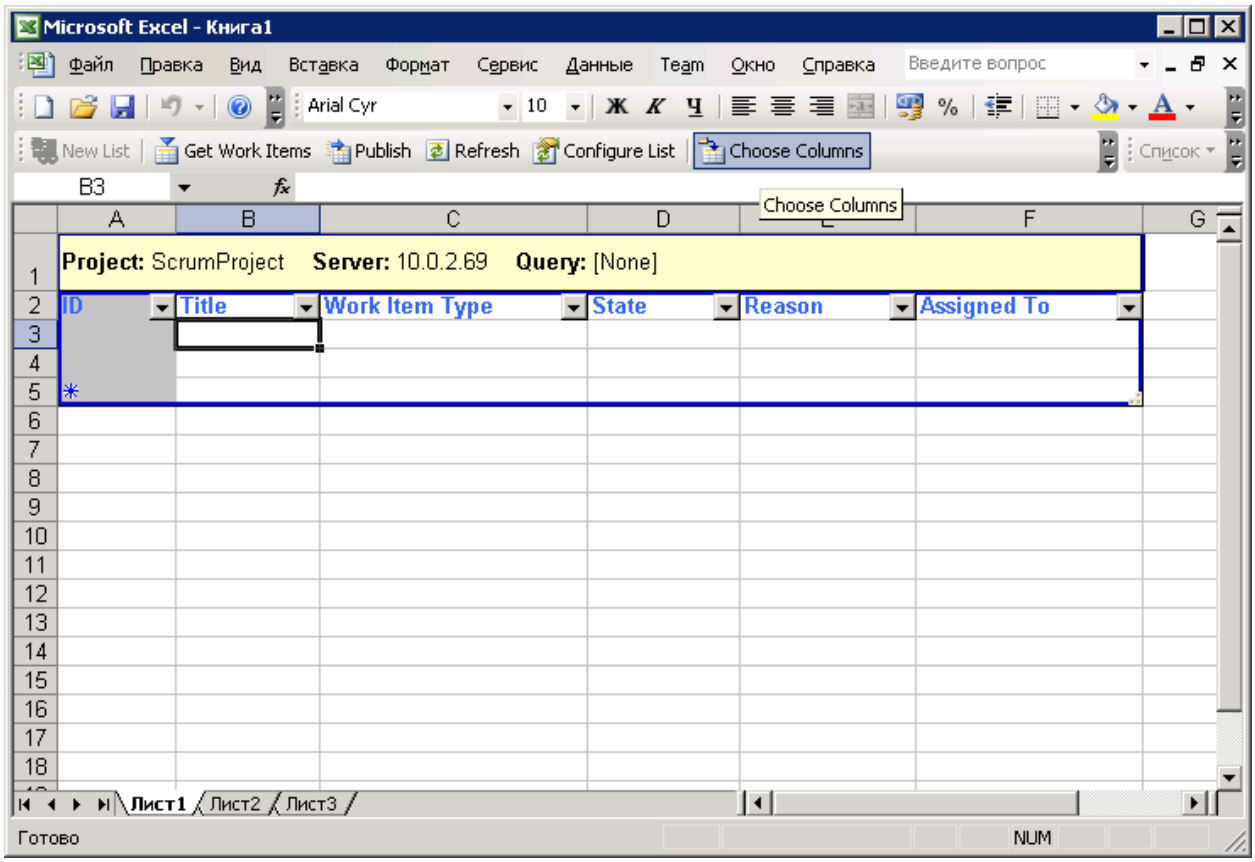
Шаг 1. Импорт списка пользовательских историй.

Для того, чтобы загрузить пользовательские истории из Excel-файла, полученного командами на прошлом занятии нужно:

1. Выбрать команды Add work items using Microsoft Excel в контекстном меню:



2. В открывшемся окне Excel настроить колонки таким образом, чтобы они совпадали порядком и смыслом с колонками в исходном Excel документе (для этого можно использовать команды Choose columns):

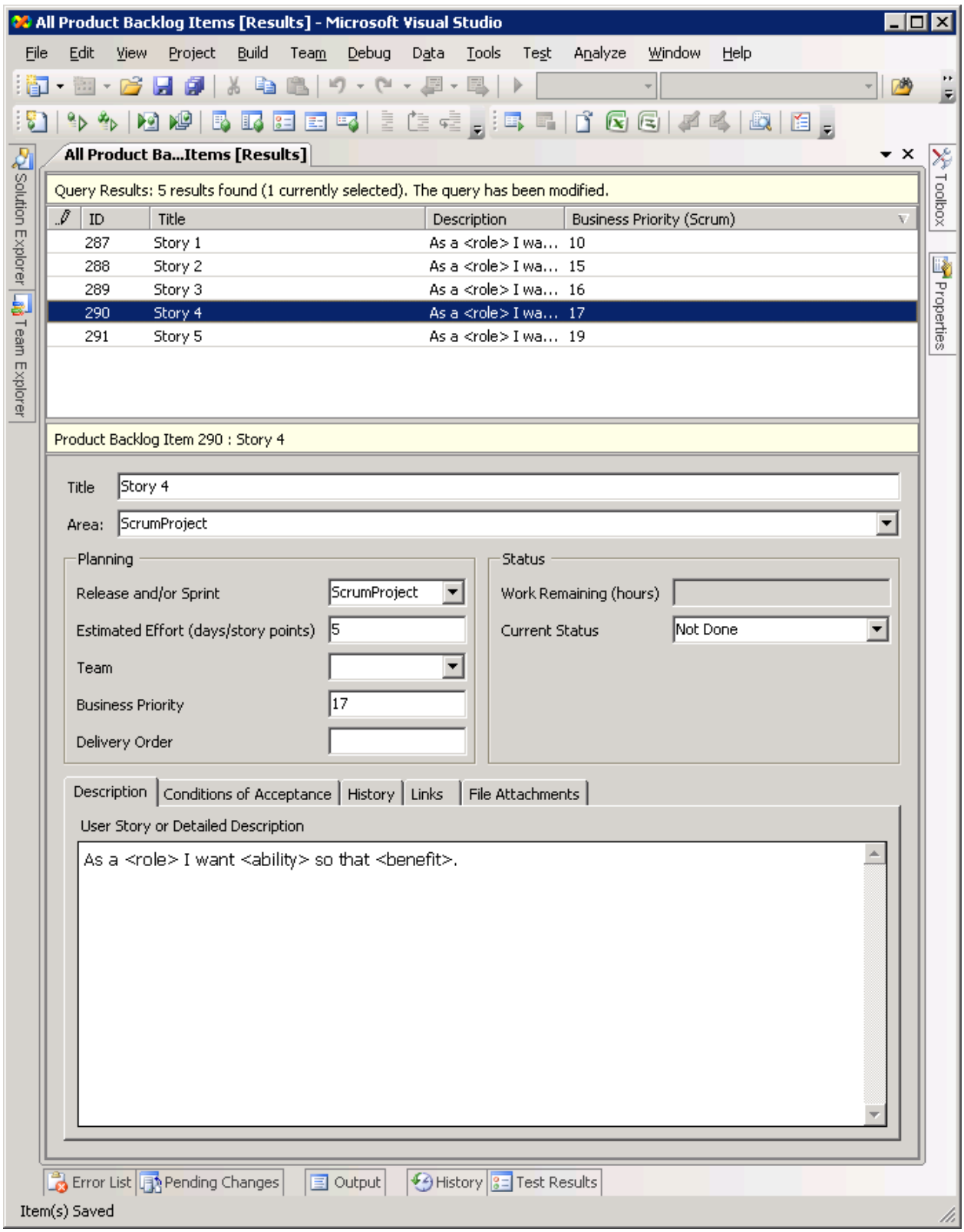


3. После того, как колонки настроены, скопировать значения из исходного Excel в редактируемый.

4. Показать колонку с именем Work Item Type и задать для все строчек значение Product backlog item.

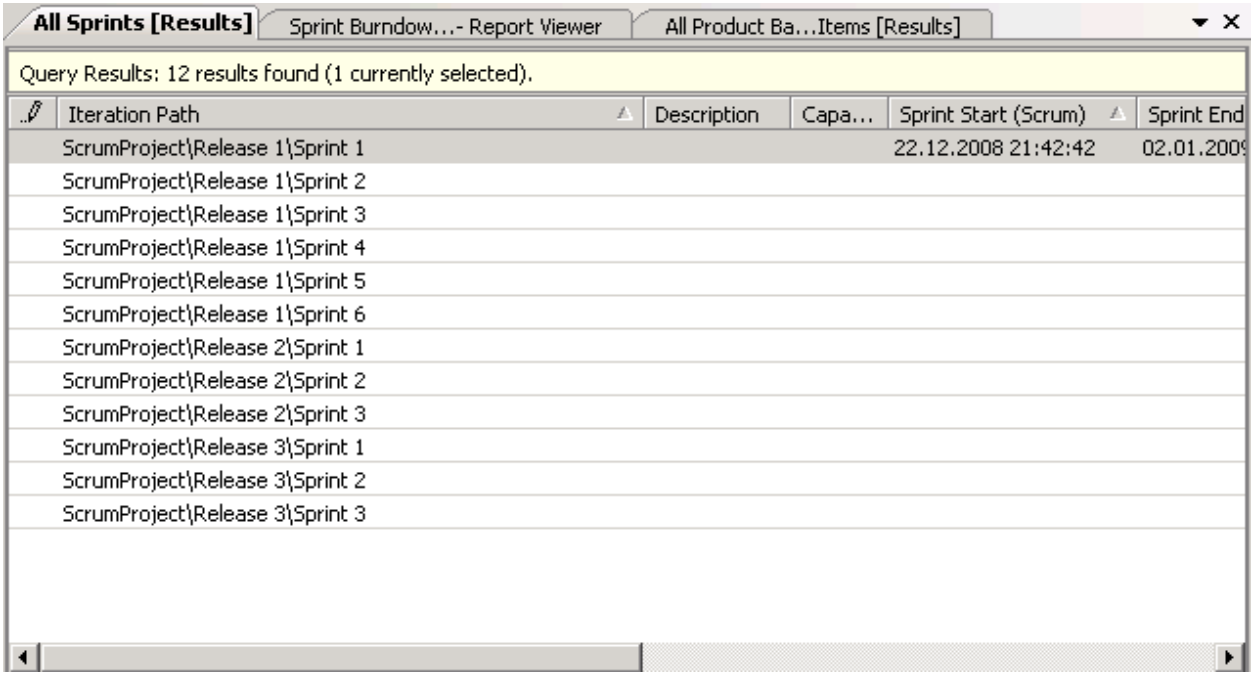
5. Нажать кнопку Publish.

6. Убедится, что при выполнении запроса All product backlog items, видны все вновь загруженные пользовательские истории:



Шаг 2. Создание sprint

После импорта списка пользовательских историй команда должна создать элемент работы, соответствующий начинающемуся sprint. Некоторое количество sprints уже создано по умолчанию при создании проекта:



Для активации первого спринта ему необходимо установить дату начала, дату окончания и количество часов, которые команда может потратить в этом спринте.

Шаг 3. Формирование Sprint backlog

После обсуждения открытых вопросов по 10 наиболее приоритетным пользовательским историям команда должна приступить к планированию текущего sprint и формированию sprint backlog. Для этого ей необходимо рассмотреть список всех пользовательских историй и разбить его на список более мелких задач. При этом для каждой задачи необходимо создать элемент работы типа sprint backlog item и проставить следующие атрибуты:

1. В качестве sprint указать Release1/Sprint1.
2. Добавить связь с соответствующим элементом product backlog, а также со всеми связанными элементами работы.
3. Установить Estimated efforts и Work remaining в соответствии с оценкой команды.
4. Задать ответственного за задачу (Owned By).

Выполнить все операции нужно средствами Visual Studio и Team Explorer.

После создания и распределения задач каждый член команды должен на своей машине убедиться, что выданные ему задачи отображаются в результатах запроса My Sprint Backlog Items.

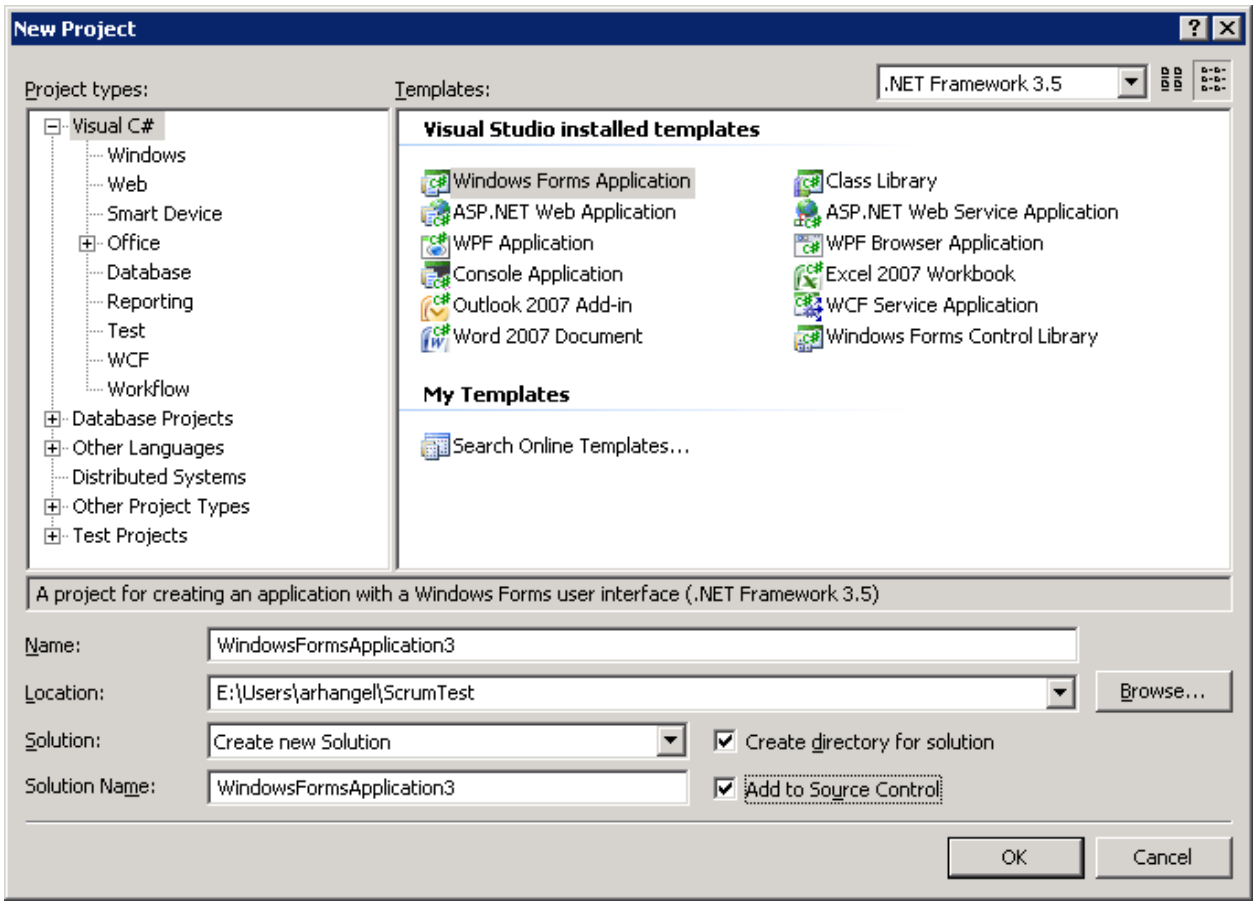
Тема 3. Работа с системой контроля версий

Основной целью данного занятия является освоение системы контроля версий Team Foundation Server и её интеграции с системой отслеживания задач. Занятие предполагает выполнение следующих действий.

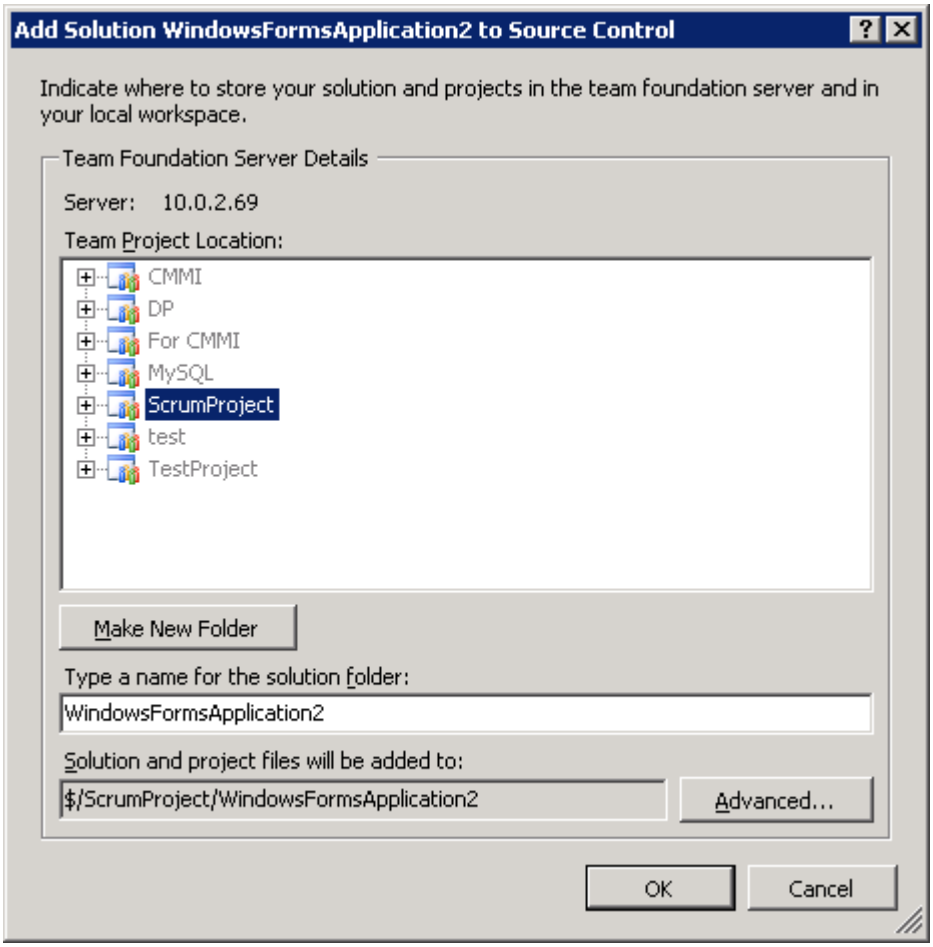
1. Разработка кода модельной задачи средствами Visual Studio и внесение его в систему управления версиями.
2. Проставление связей между вносимыми изменениями и элементами системы отслеживания задач.
3. Создание параллельно поддерживаемых веток кода.
4. Интеграция изменений, сделанных параллельно в одном файле или в разных ветках кода.

Шаг 1. Разработка кода.

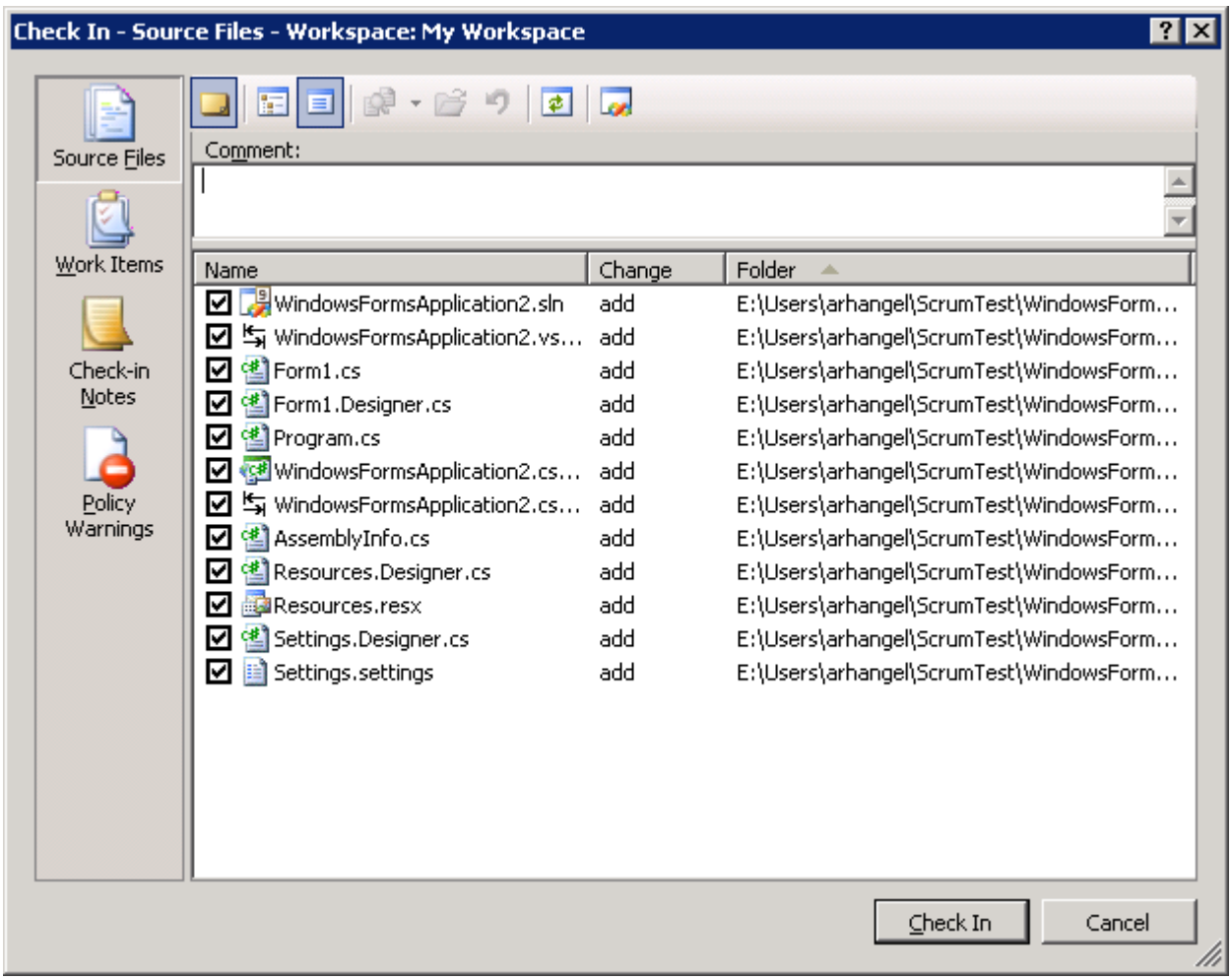
Перед началом работы команде необходимо создать решение (solution) средствами Visual Studio, включив опцию Add to Source Control:



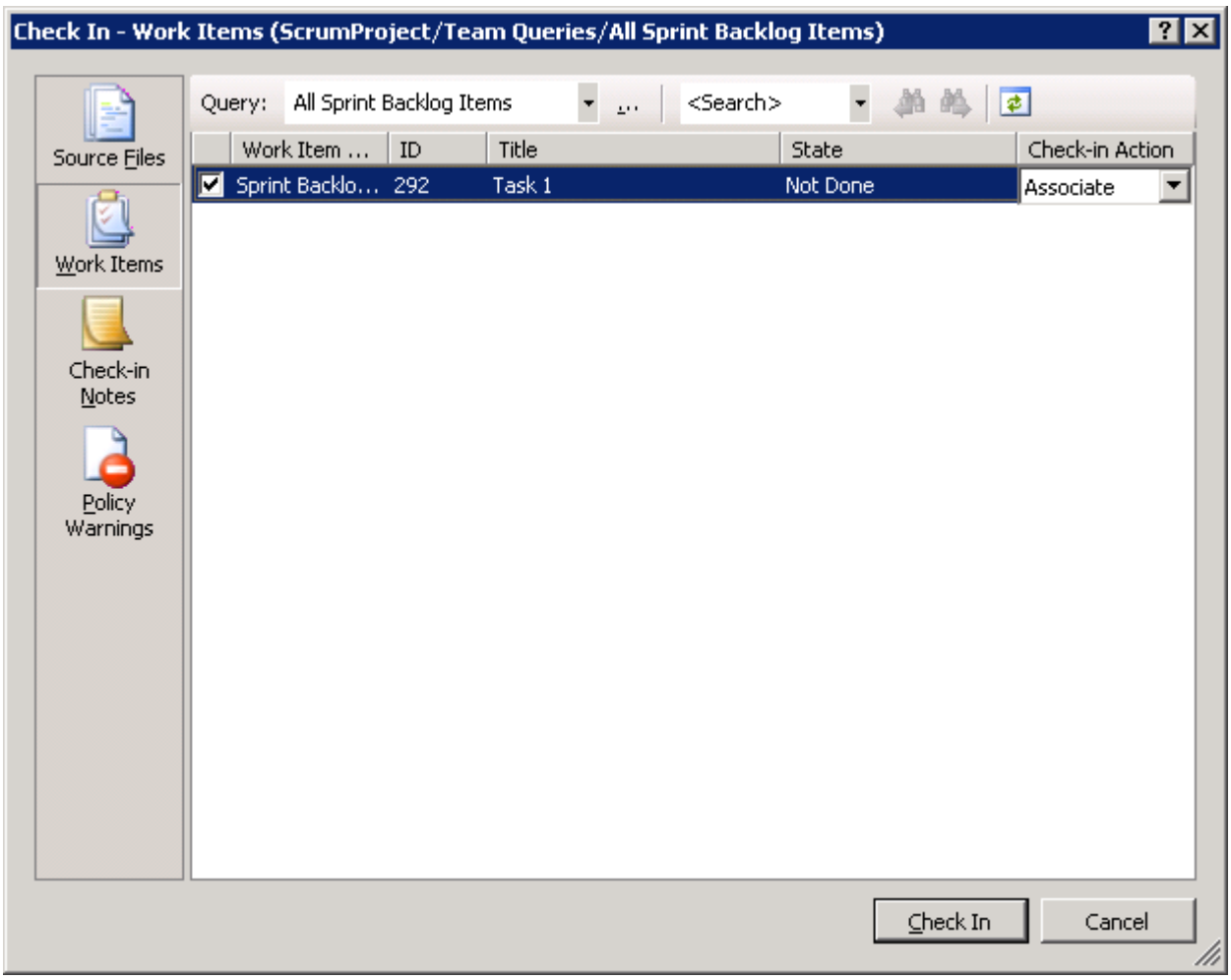
В открывшемся после создания проекта окне необходимо выбрать командный проект, в систему контроля версий которого нужно добавить данное решение:



Затем необходимо внести все данные в систему контроля версий, используя команду Check-in, открывающую диалог:



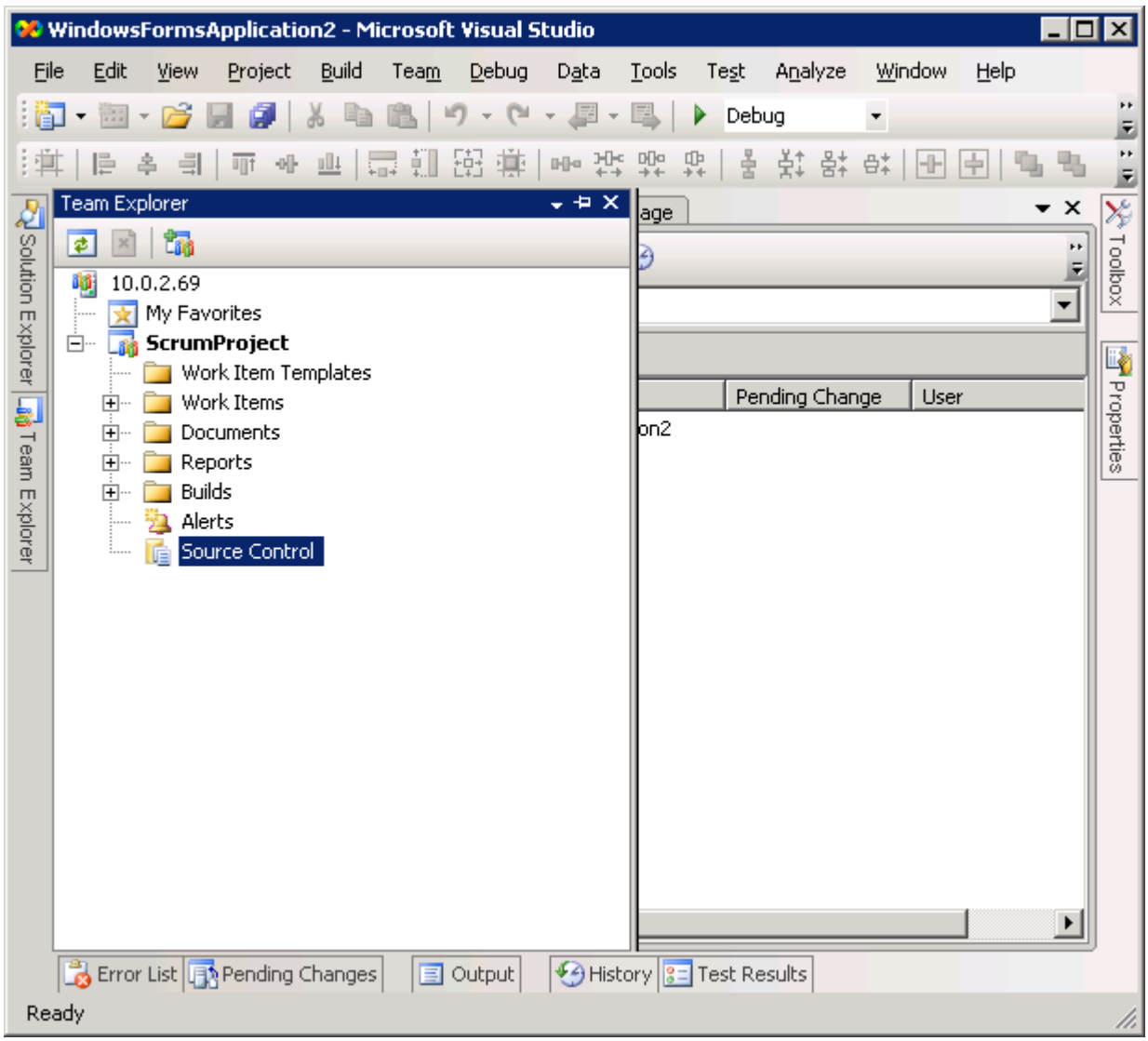
В этом диалоге необходимо внести комментарии к вносимому коду, а также, на вкладке Work items, связать вносимое изменение с элементами работы:



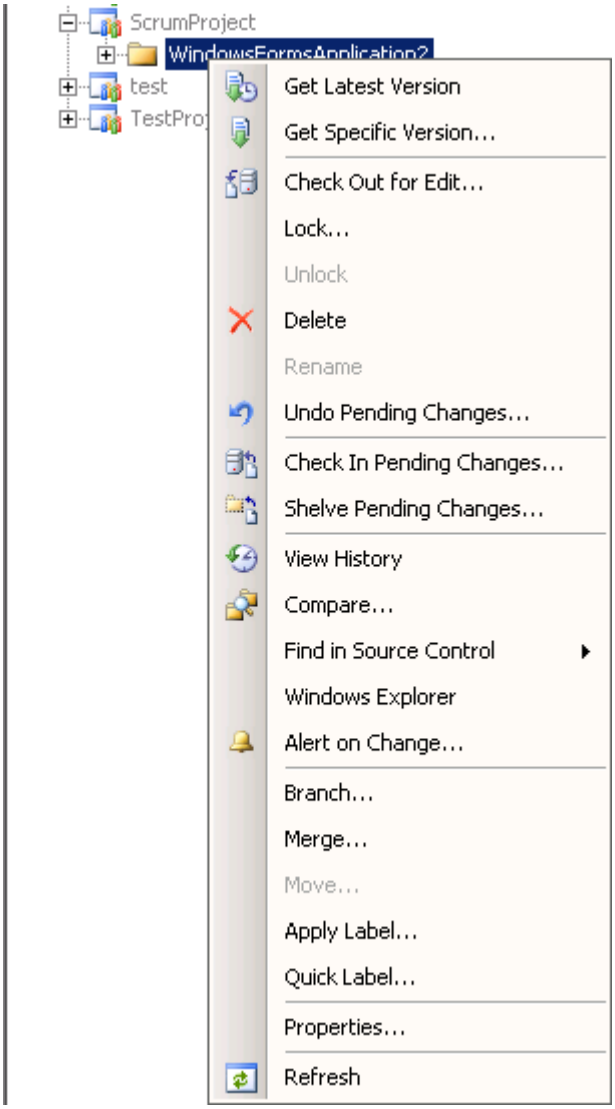
Шаг 2. Создание ветки кода.

Для того, чтобы освоится с практикой конфигурационного управления, команды должны создать ветвь в системе контроля версий, следуя приведенной ниже инструкции.

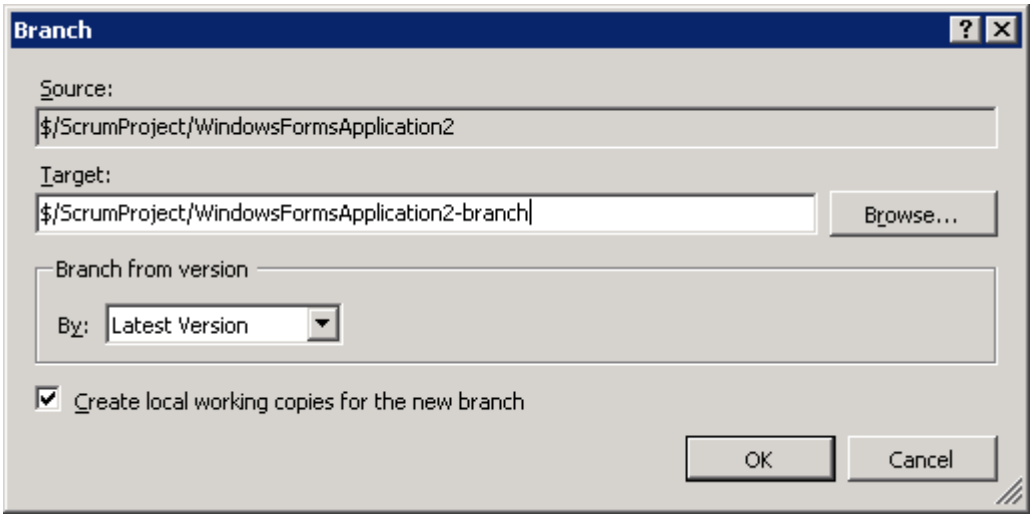
1. Открыть Source control explorer:



2. Выбрать нужный проект и в контекстном меню команду Branch:



3. В открывшемся окне задать целевую папку, куда необходимо скопировать данные для новой ветви:

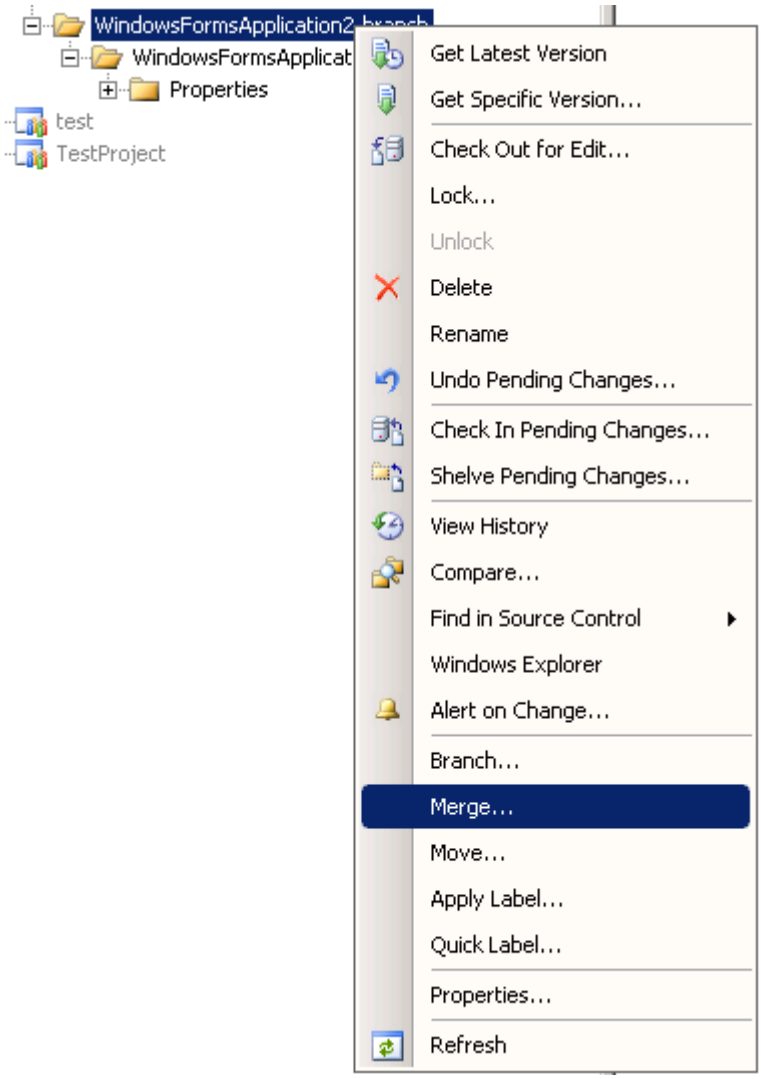


4. Внести изменения с помощью команды Check-in

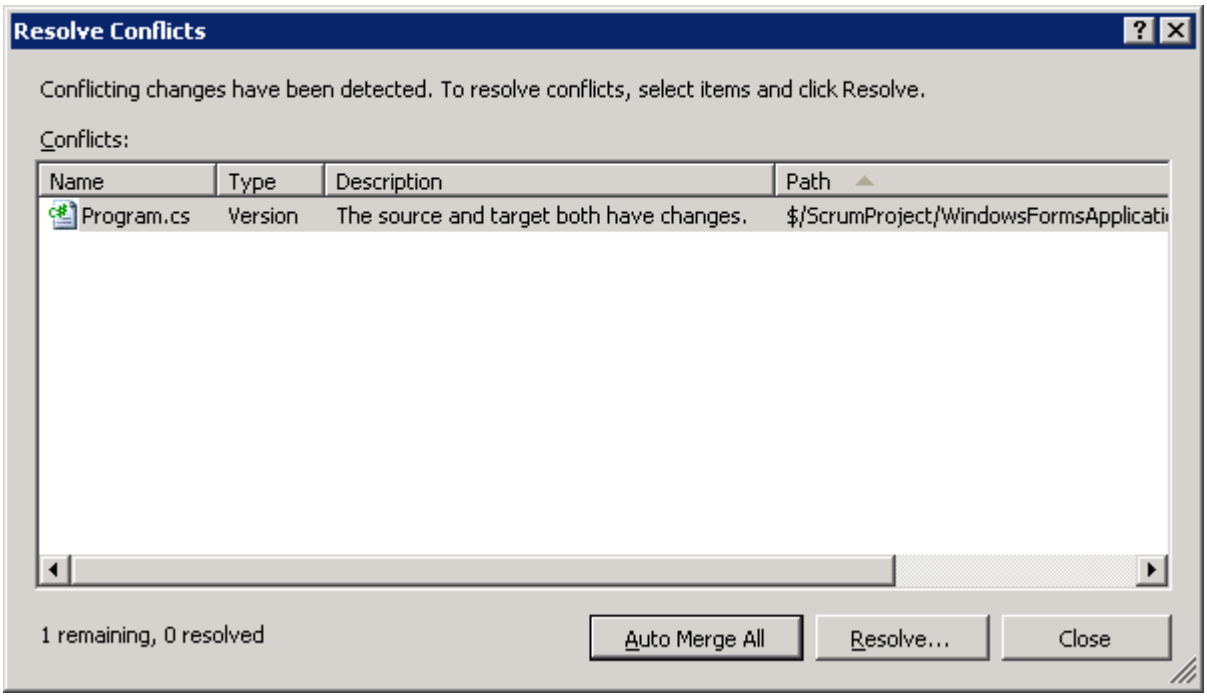
После того, как создана ветка, разные участники команды вносят изменения в разные ветки кода, реализую необходимую функциональность приложения.

Шаг 3. Объединение изменений

После того, как в отдельные ветви было внесено некоторое количество изменений, необходимо перенести изменения из отделенной ветви в основную, используя команду Merge:



В процессе объединения изменений могут возникнуть конфликты, информация о которых будет включена в сообщение следующего вида:



Все конфликты необходимо разрешить, используя команду Resolve и утилиту для объединения результатов.

После разрешения конфликтов все изменения внести в систему контроля версий посредством операции Check-in.

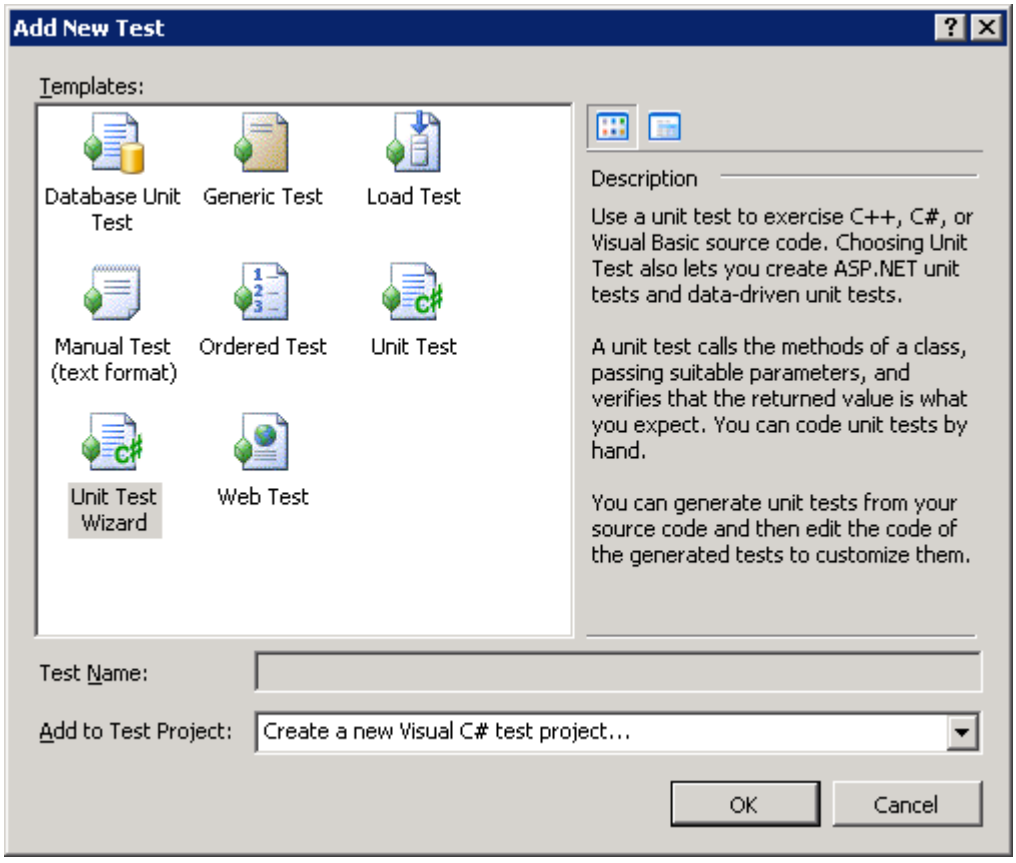
Тема 4. Разработка модульных тестов.

На данном занятии команды должны разработать набор модульных тестов, покрывающих функциональность, разработанную на занятии предыдущем. В рамках данного занятие предполагается освоить следующие возможности MS VSTS.

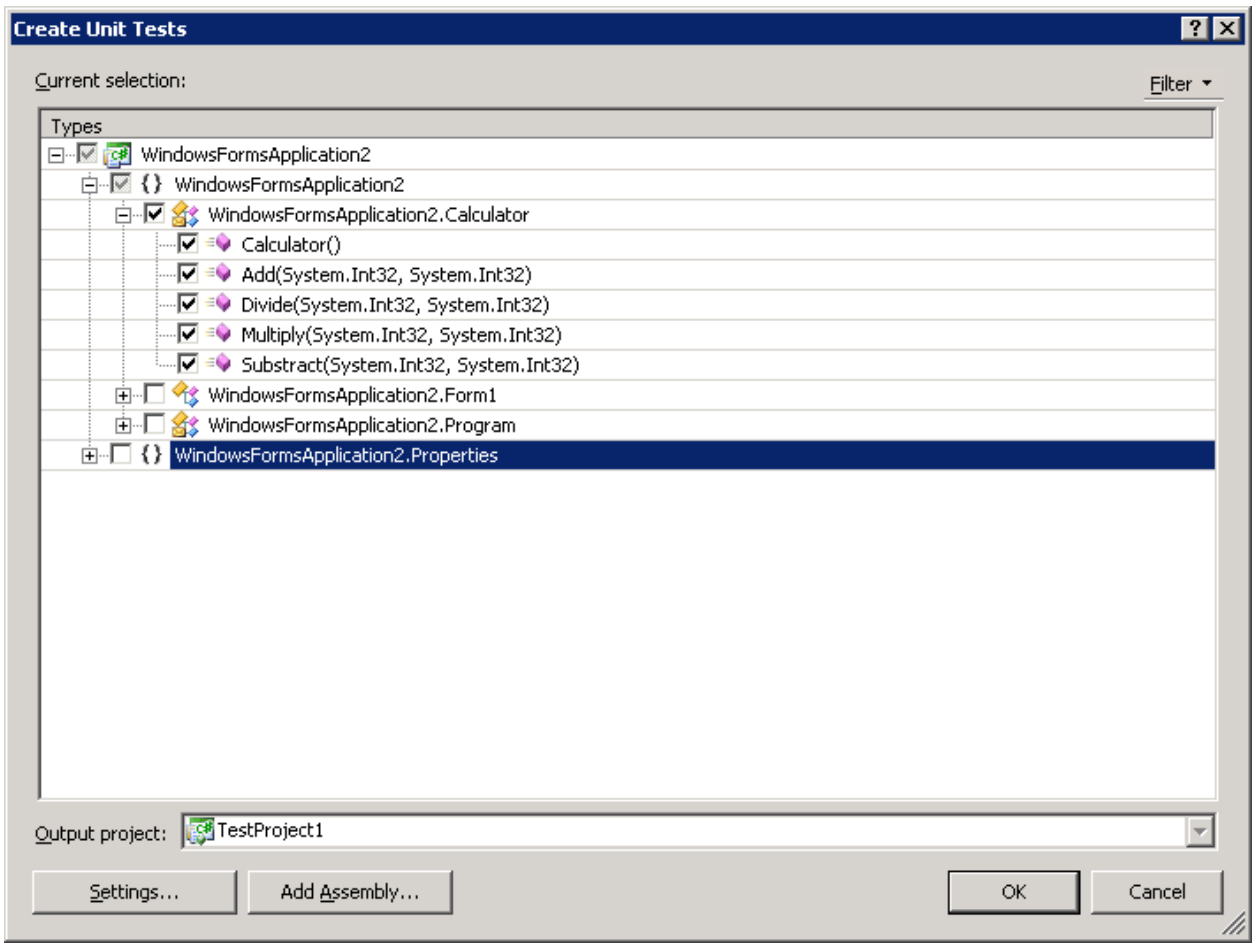
1. Автоматическая генерация тестов.
2. Наполнение тестов содержимым.
3. Запуск тестов и просмотр результатом.
4. Изменение конфигурации работы тестов.

Шаг 1. Автоматическая генерация тестов.

Для ускорения разработки тестов команды могут воспользоваться возможность Visual Studio по автоматической генерации тестов. Для этого необходимо воспользоваться командой Test/New Test и выбрать Unit Test wizard в открывшиеся окне:



После создания тестового проекта, будет предложен выбор из тех типов и методов, для тестирования которых необходимо создать заглушки:



В этом диалоге команда должна выбрать все основные классы и методы, которые планируется покрыть модульными тестами.

Шаг 2. Наполнение тестов содержимым.

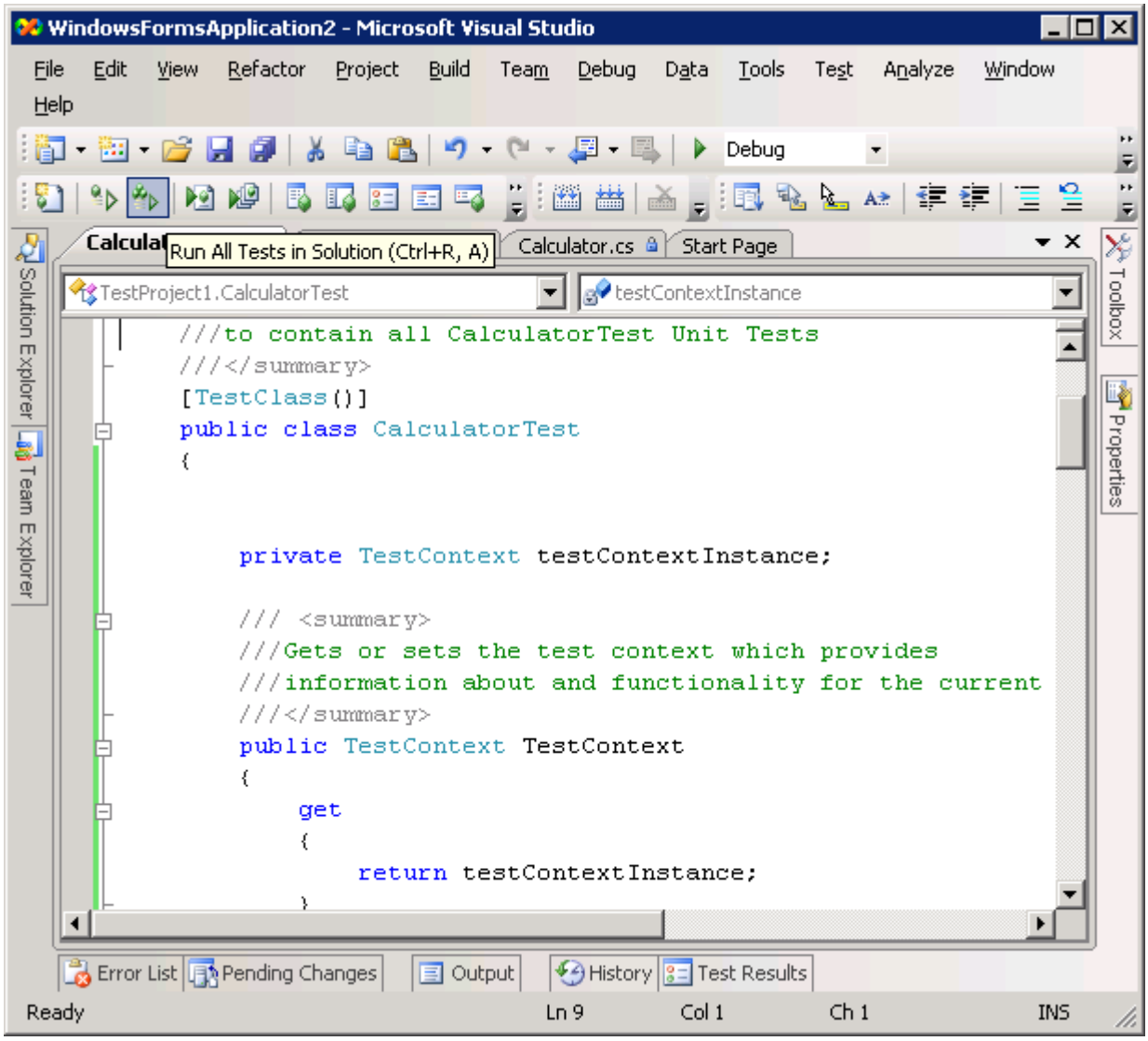
После генерации тестового покрытия команда получить набор скелетов тестов для всех методов, которые были выбраны для тестирования. Однако, эти тесты имеют достаточно простую структуру и пока лишены смысла:

```
/// <summary>
///A test for Multiply
///</summary>
[TestMethod()]
public void MultiplyTest()
{
    // TODO: Initialize to an appropriate value
    Calculator target = new Calculator();
    int a = 0; // TODO: Initialize to an appropriate value
    int b = 0; // TODO: Initialize to an appropriate value
    int expected = 0; // TODO: Initialize to an appropriate value
    int actual;
    actual = target.Multiply(a, b);
    Assert.AreEqual(expected, actual);
    Assert.Inconclusive("Verify the correctness of this test
    method.");
}
```

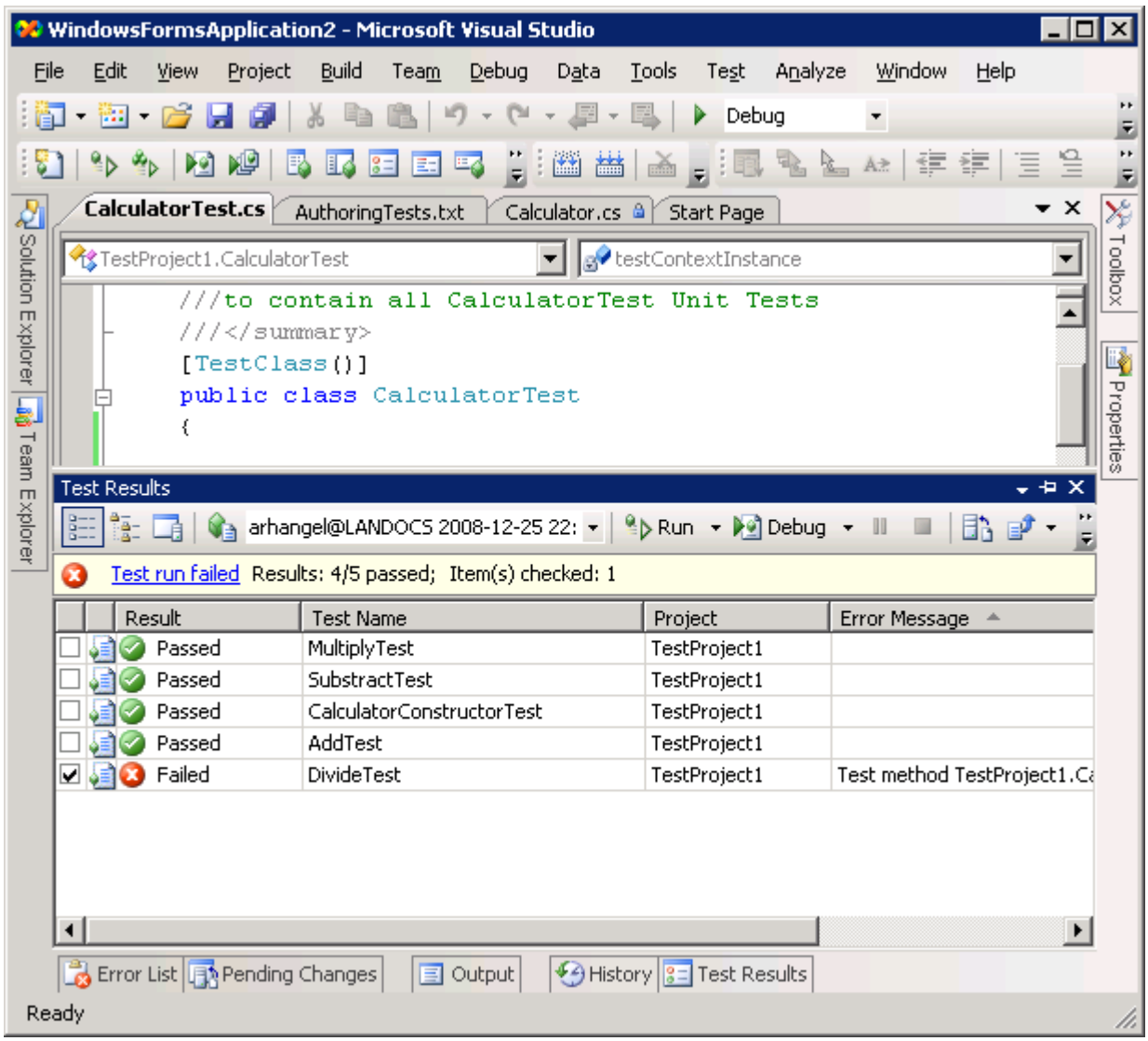
На следующем шаге команда должна заполнить эти тесты необходимым содержимым, используя функциональность по валидации (Assert), предоставляемую тестовой платформой.

Шаг 3. Запуск тестов

Для того, чтобы исполнить созданные тесты необходимо использовать соответствующую панель инструментов:



После этого результаты выполнения тестов будут видны в окне результатов:

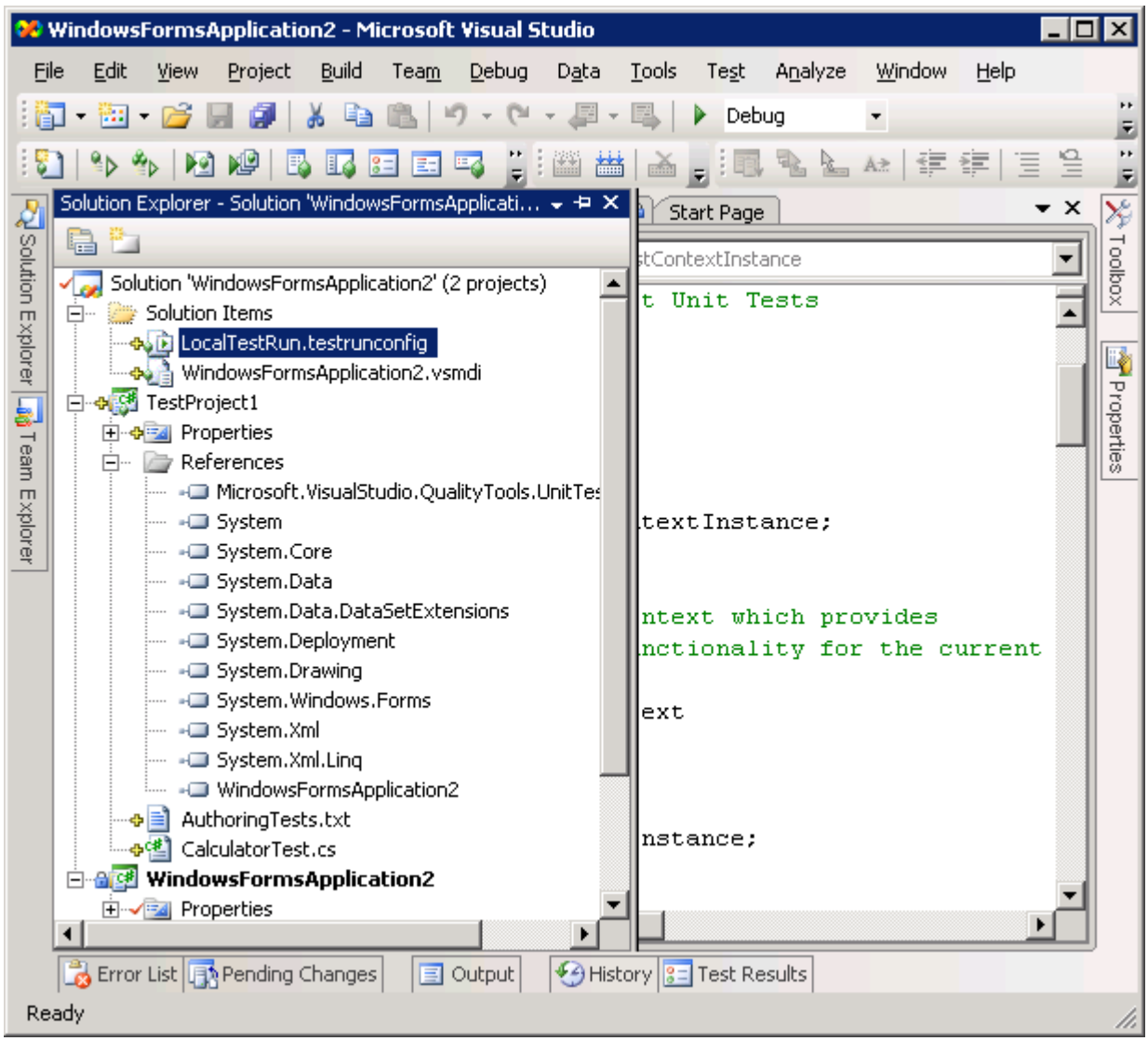


Команды должны добиться того, чтобы все разработанные тесты проходили успешно.

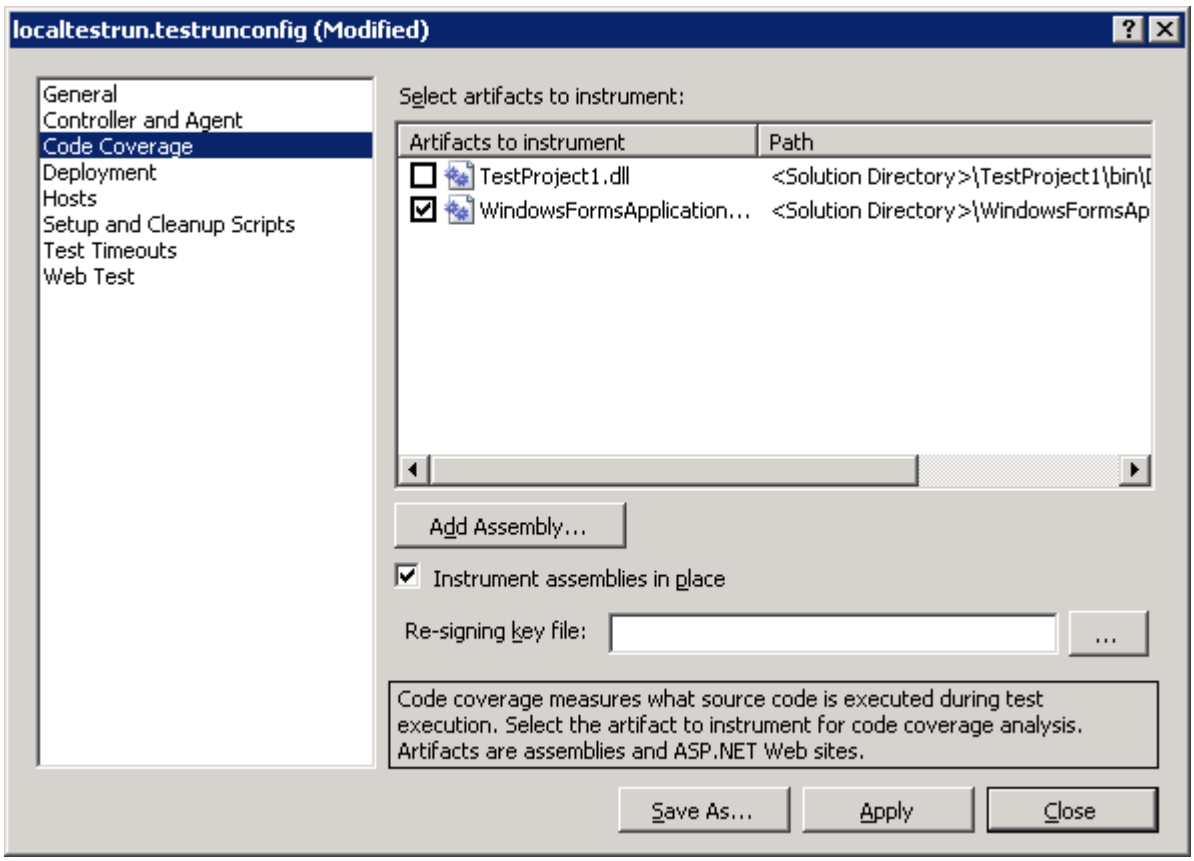
Шаг 4. Изменение конфигурации тестов.

Для того, чтобы проанализировать качество разработанных тестов, команда должна вычислить тестовое покрытие. Для этого её необходимо:

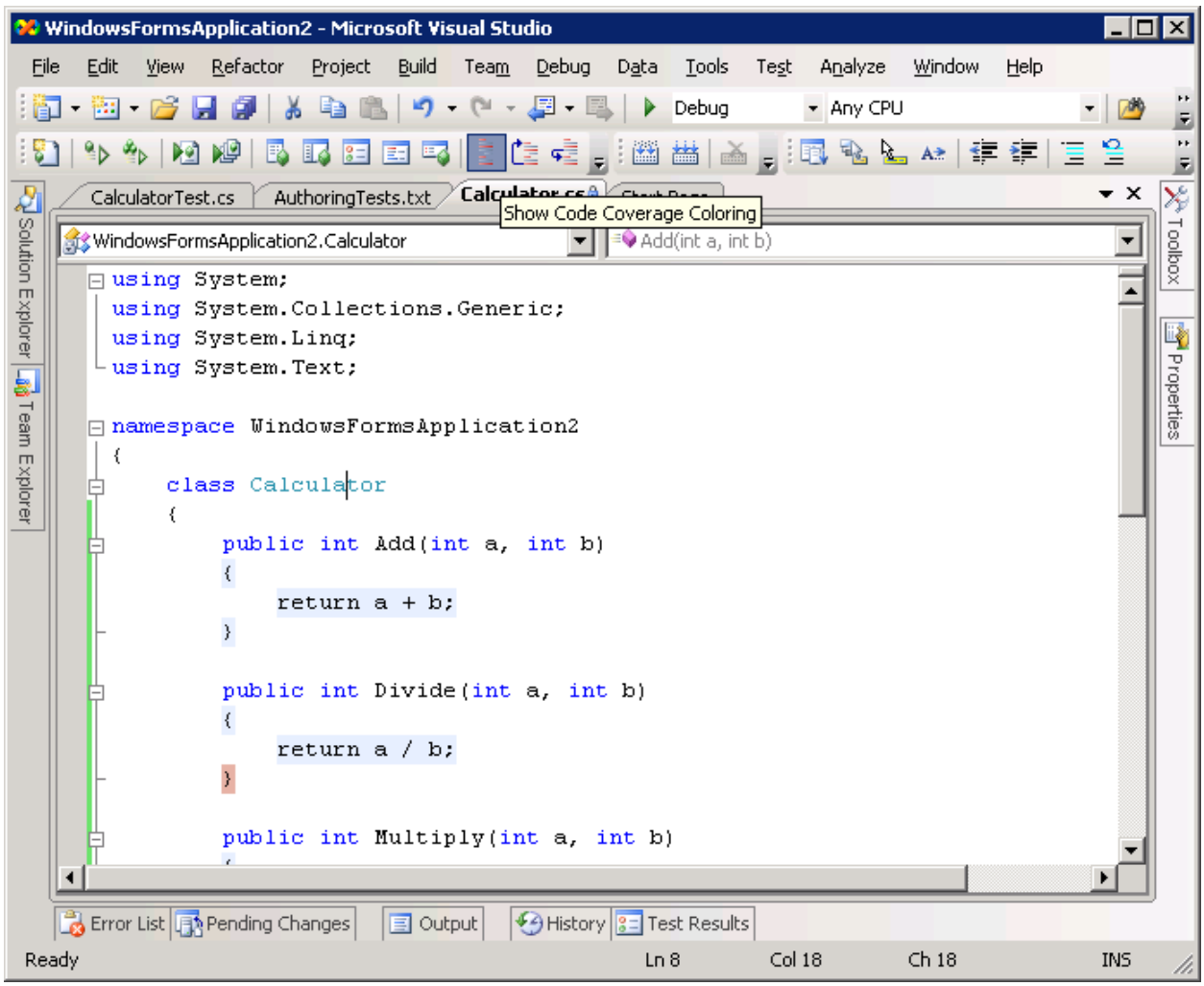
1. Открыть файл конфигурации запуска тестов, автоматически добавленный к решению при создании тестов:



2. На открывшемся диалоге выбрать вкладку Code Coverage и установить то, какие именно проекты нужно анализировать:



3. После сохранения конфигурации запустить тесты и активировать опцию Show Code Coverage Coloring:


WindowsFormsApplication2 - Microsoft Visual Studio
File Edit View Refactor Project Build Team Debug Data Tools Test Analyze Window Help
Debug
Any CPU
CalculatorTest.cs
AuthoringTests.txt
Show Code Coverage Coloring
WindowsFormsApplication2.Calculator
Add(int a, int b)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace WindowsFormsApplication2
{
class Calculator
{
public int Add(int a, int b)
{
return a + b;
}
public int Divide(int a, int b)
{
return a / b;
}
public int Multiply(int a, int b)
Solution Explorer
Team Explorer
Toolbox
Properties
Error List
Pending Changes
Output
History
Test Results
Ready
Ln 8
Col 18
Ch 18
INS

Тема 5. Создание и конфигурация автоматической сборки

На данном этапе команда должна создать в своем проекте процедуру автоматической сборки. При этом должно быть создано несколько процедур:

1. Простая процедура, включающая только сборку.

2. Полная процедура, включающая тесты и анализ кода.

После создания сборок необходимо настроить параметры непрерывной интеграции:

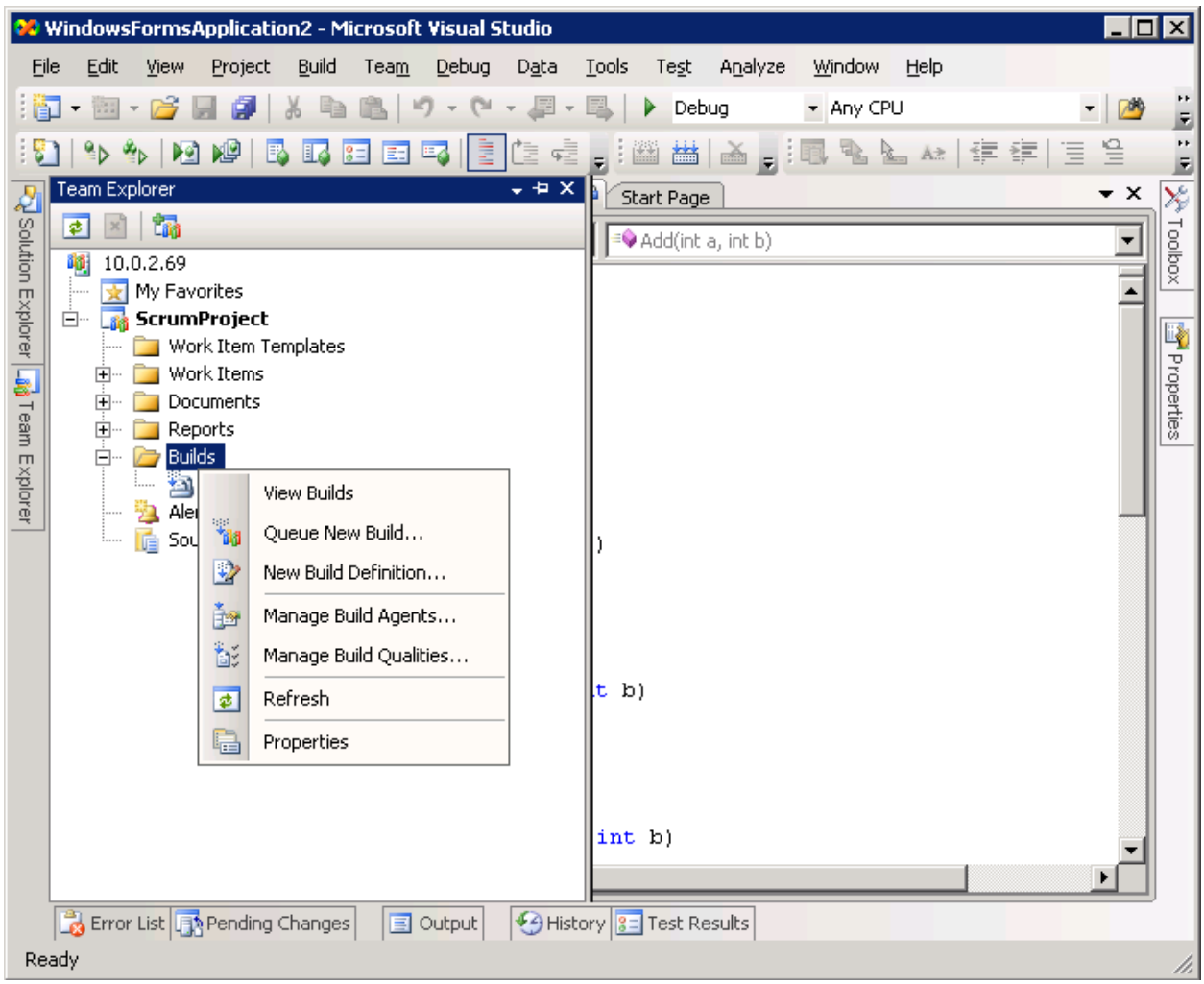
1. Простая сборка должна запускаться после каждого внесенного изменений, но не чаше чем раз в 5 минут.

2. Полная сборка должна запускаться каждую ночь.

Шаг 1. Создание простой сборки

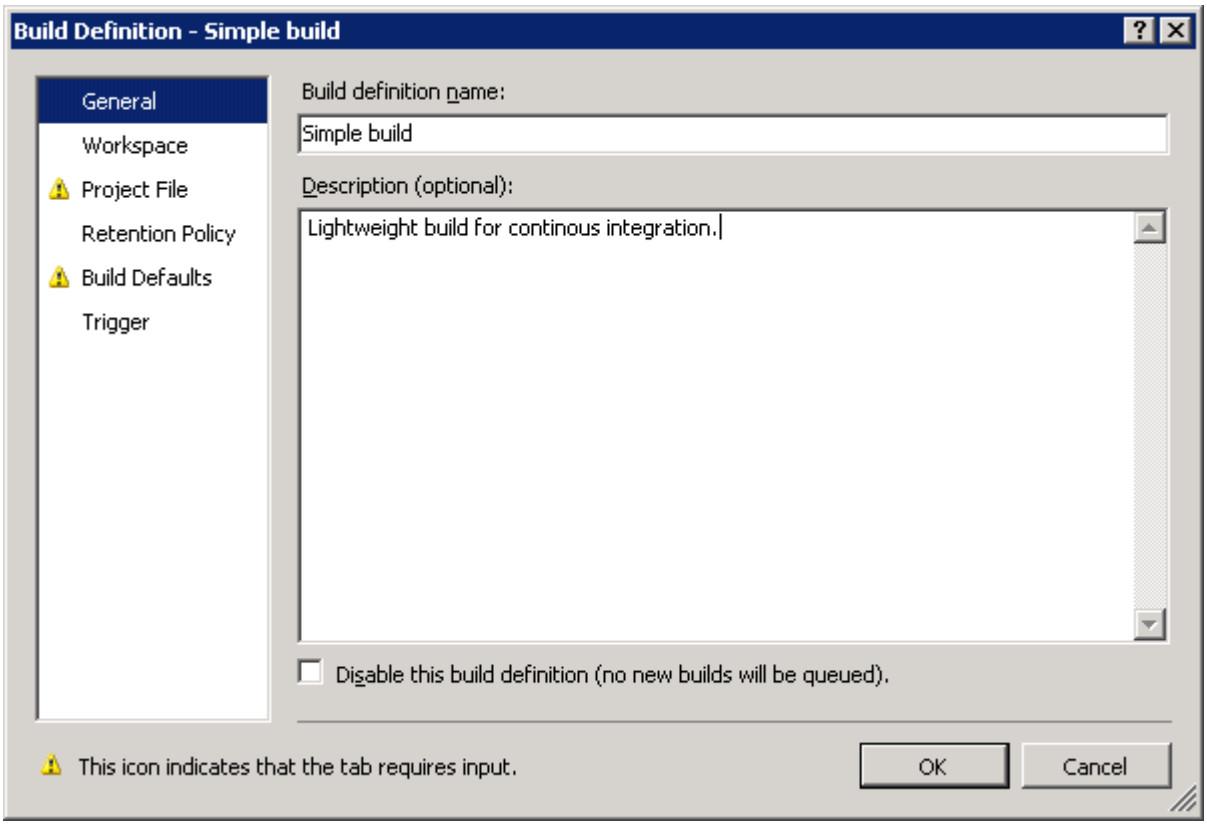
Все результаты сборки, проведенной TFS, выкладываются в разделяемую папку, на запись в которую есть права у пользователя, с правами которого работает сервер автоматических сборок (обычно – TFSBuild), а также у пользователя, с правами которого работает сам TFS (обычно – TFSService). Как правило, учащиеся не обладают достаточным количеством прав для создания такого рода папок, поэтому они должны быть заранее подготовлены преподавателем.

Для создания простой сборки необходимо обратится к окну Team Explorer, после чего в разделе Builds выбрать команду Build Definitions:

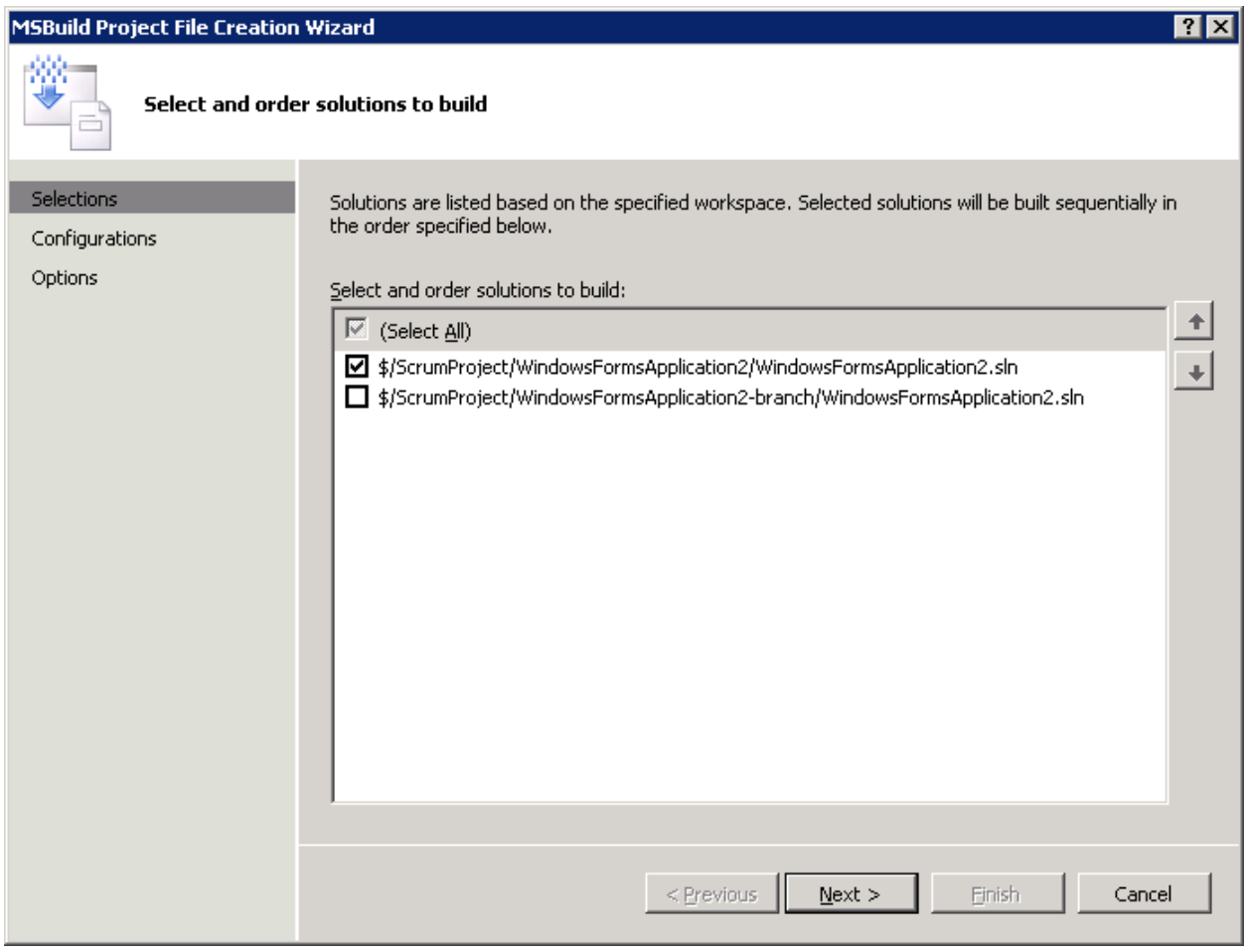


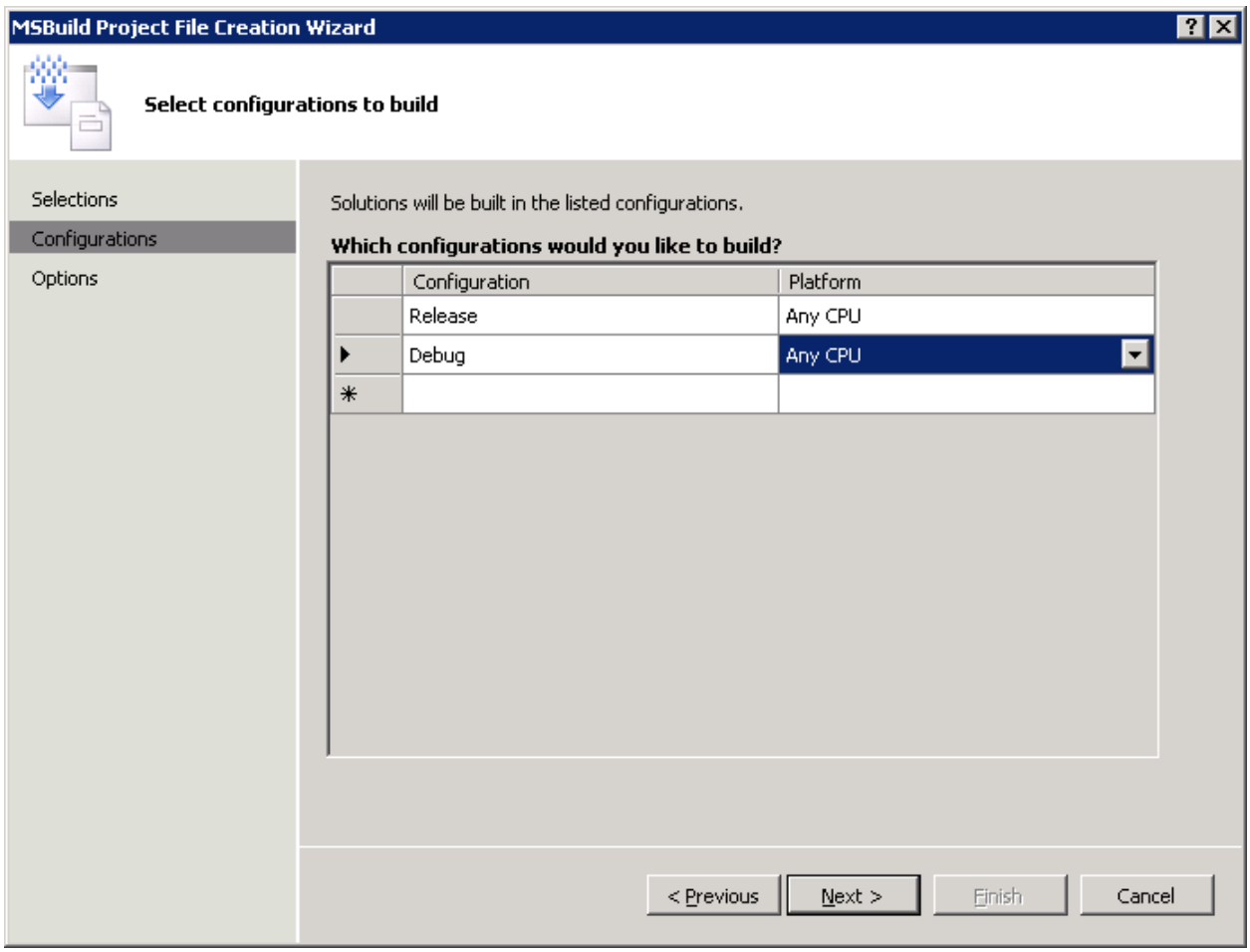
Вызов этой команды приведет к открытию мастера, в котором нужно задать следующие параметры сборки:

1. Задать имя и описание сборки:

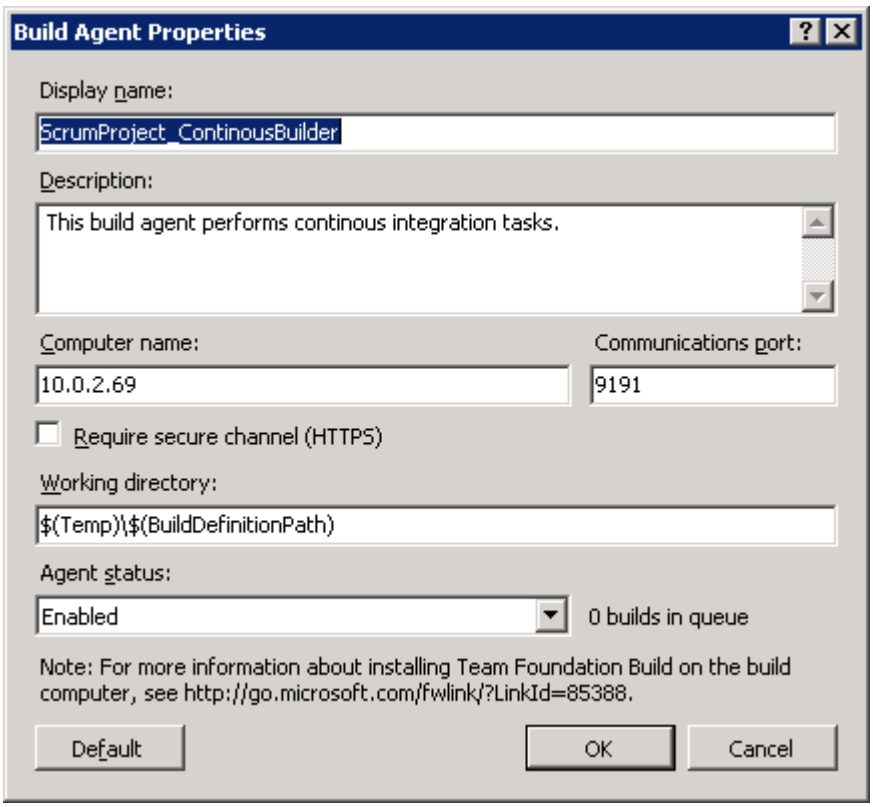


2. На закладке Project File, создать новое описание сборки используя кнопку Create, после чего задав проекты и конфигурации для сборки:

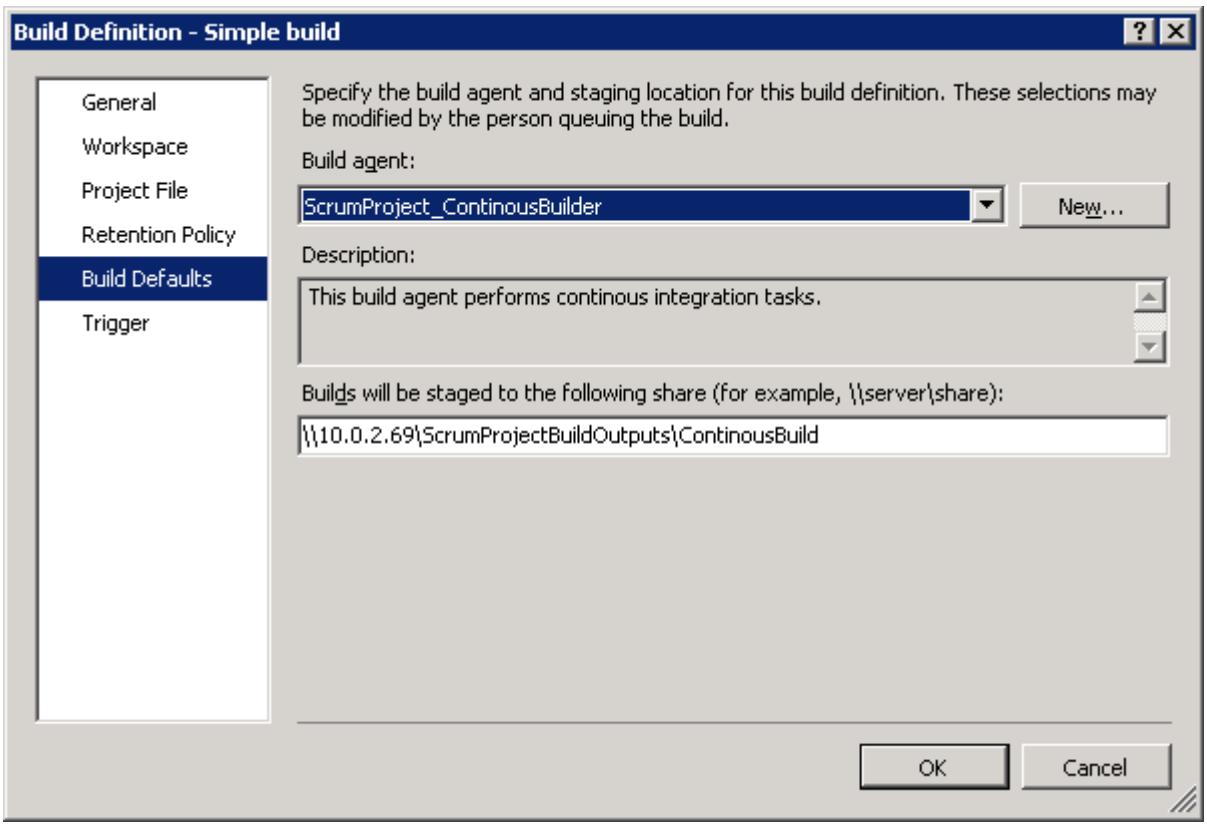

MSBuild Project File Creation Wizard
Select and order solutions to build
Selections
Configurations
Options
Solutions are listed based on the specified workspace. Selected solutions will be built sequentially in the order specified below.
Select and order solutions to build:
(Select All)
$/ScrumProject/WindowsFormsApplication2/WindowsFormsApplication2.sln
$/ScrumProject/WindowsFormsApplication2-branch/WindowsFormsApplication2.sln
< Previous
Next >
Finish
Cancel



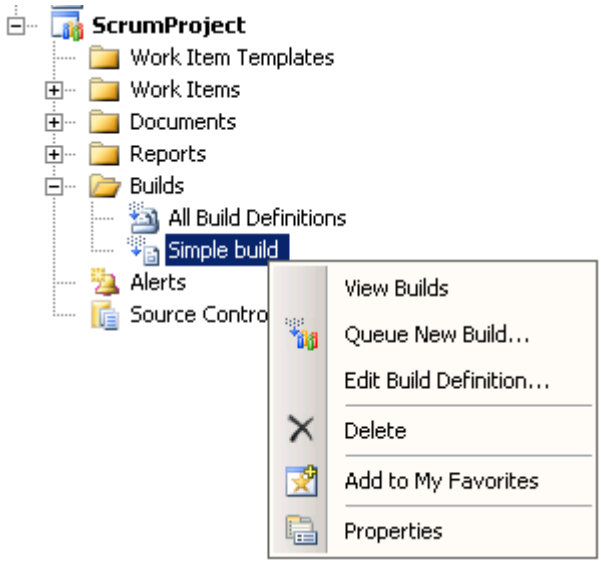
3. На закладке Build Defaults создать определение агента-сборщика используя кнопку New. Для агента указать имя, описание, и IP адрес сервера сборок (как правило, совпадает с сервером TFS):



4. На закладке Build Defaults также необходимо задать имя разделяемой папки, в которую будут сложены результаты:



После того, как определение сборки было создано, её необходимо запустить, используя команду Queue new build:

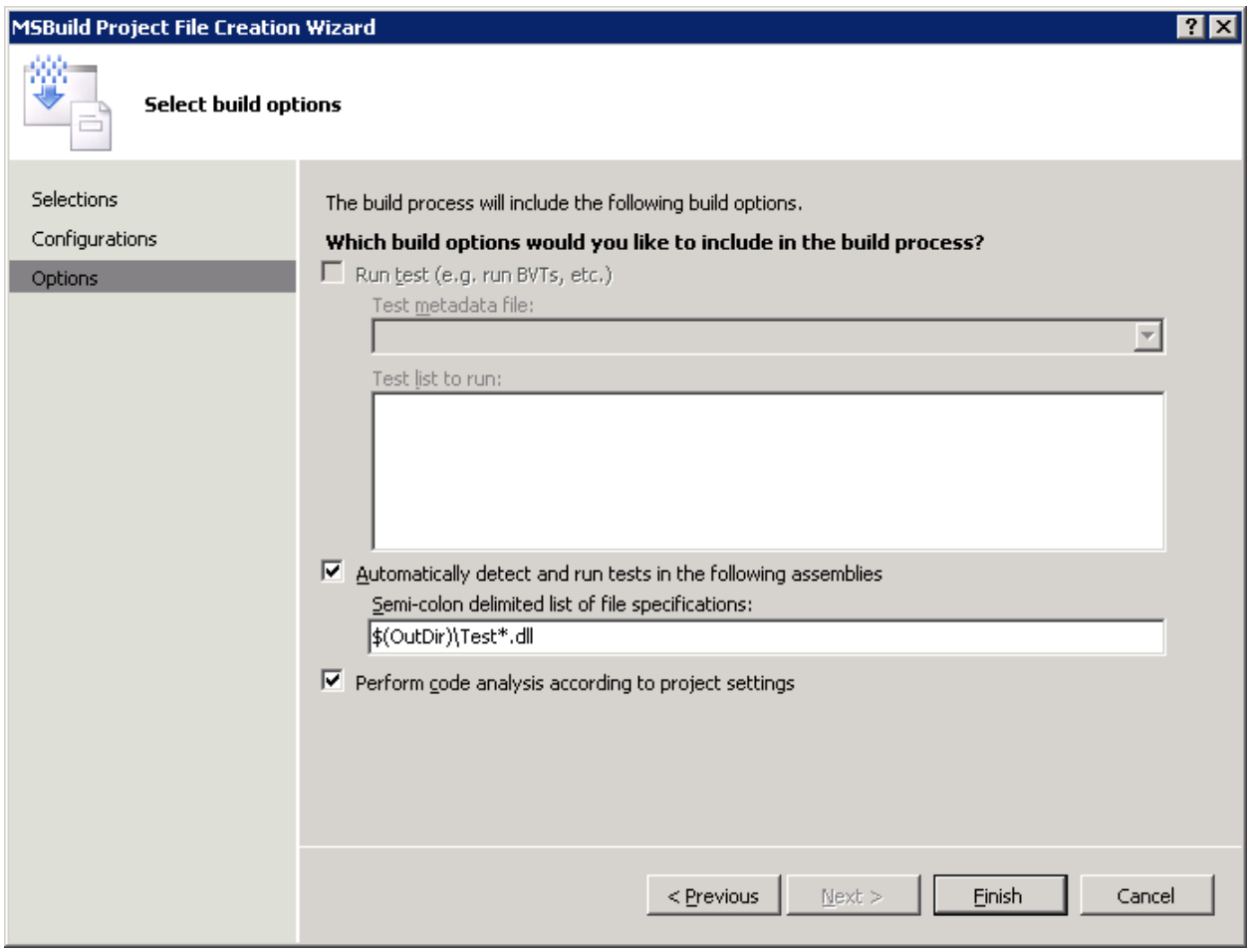


После запуска сборки необходимо дождаться ее завершения, и убедится, что на соответствующей сетевой папке появились результаты сборки.

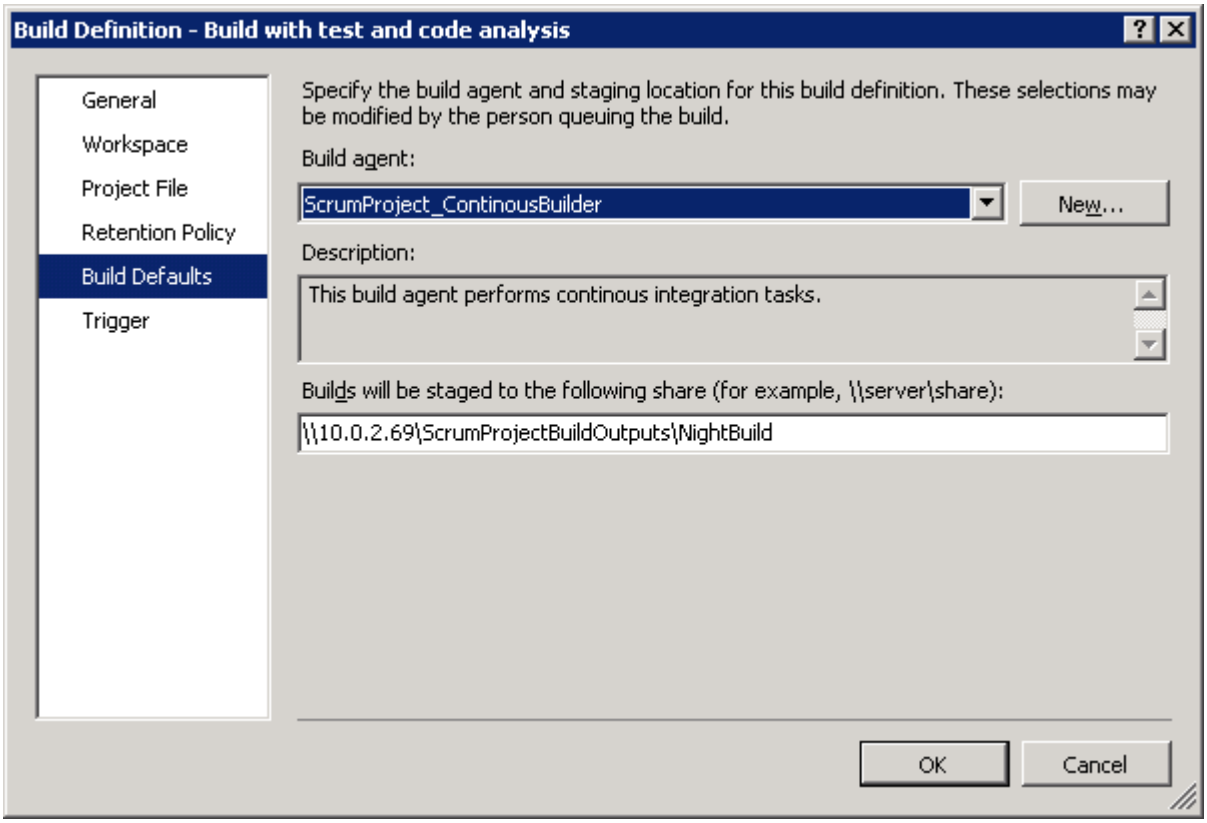
Шаг 2. Создание сложной сборки.

Создание сложной сборки проходит во многом аналогично созданию сборки простой, за исключением нескольких шагов:

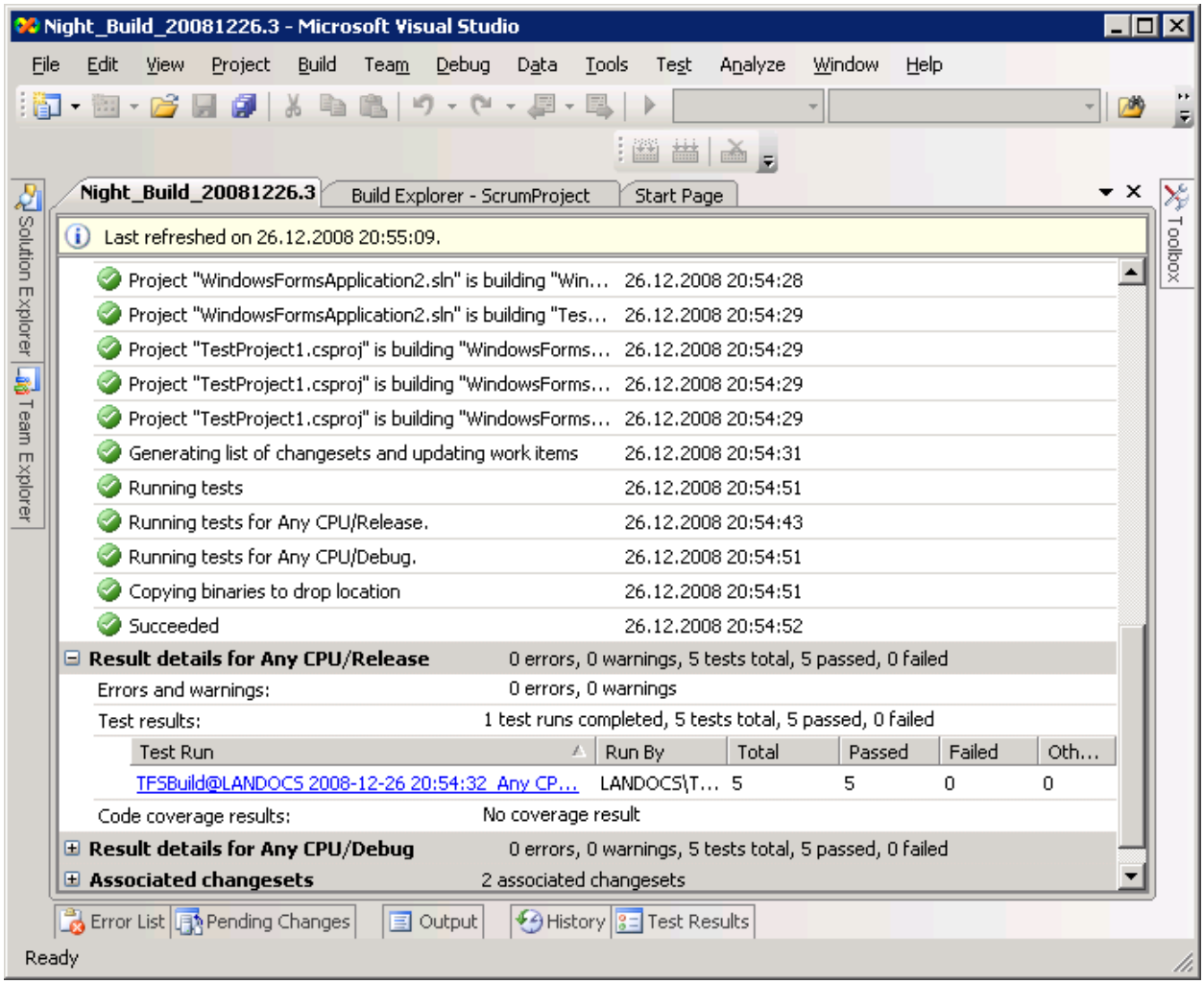
1. На закладке опций при создании проекта сборки необходимо включить автоматический запуск модульных тестов и анализ кода:



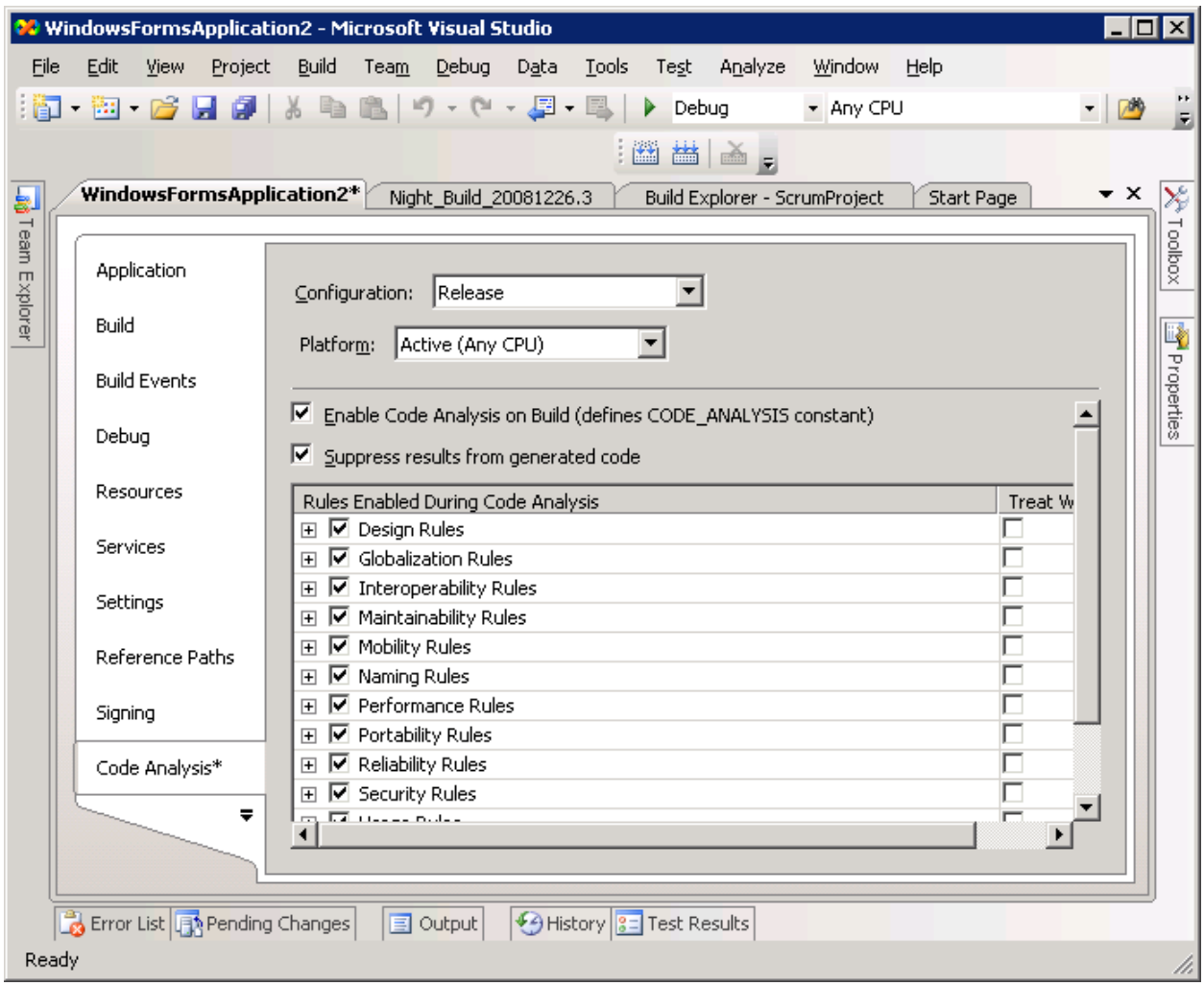
2. На закладке Build Defaults необходимо задать другую папку для сбора результатов:



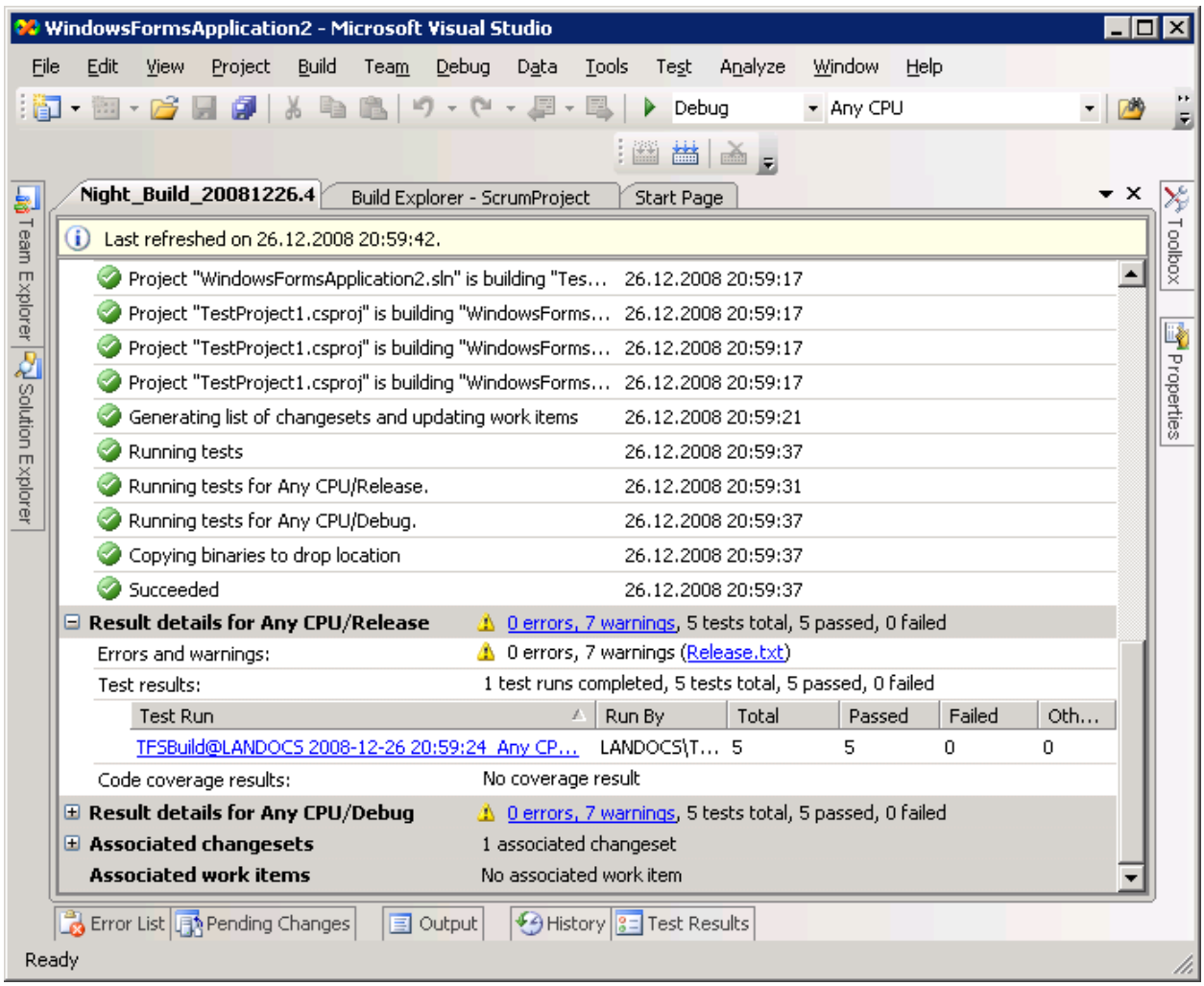
После создания сборки необходимо запустить её, и обратить внимание, что результаты сборки включают и результаты тестов:



Теперь нужно добиться выполнения статического анализа кода во время ночной сборки. Для этого необходимо активировать анализ кода в настройках соответствующих проектов:



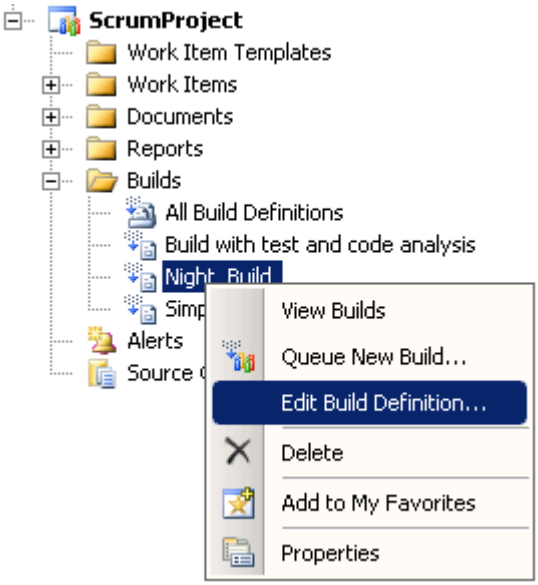
Следующая же собранная сборка будет содержать большое количество предупреждений от анализатора:



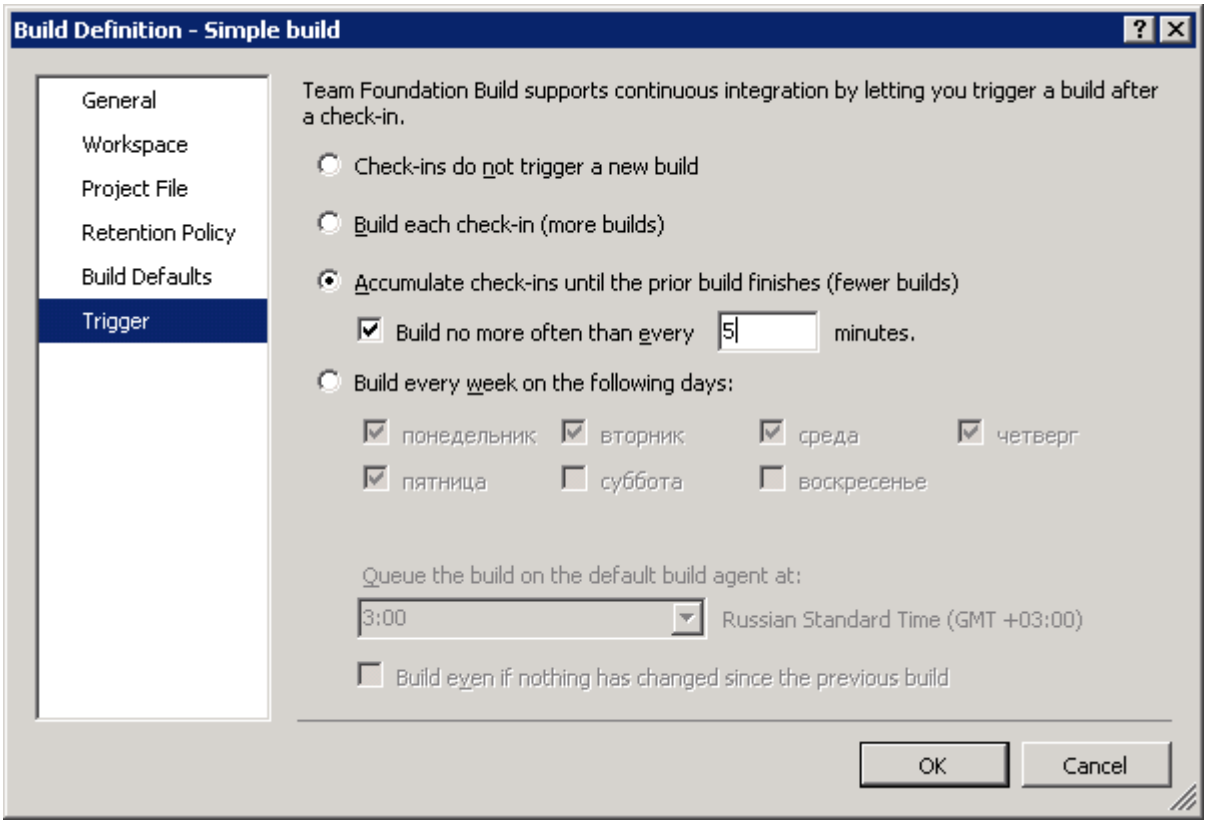
Шаг 3. Настройка непрерывной интеграции.

На данном шаге учащимся необходимо исправить описания сборок таким образом, чтобы они выполнялись автоматически при определенных условиях. Простой вариант сборки должен запускаться автоматически после каждого внесения изменений, но не чаще, чем в пять минут. Для того, чтобы добиться этого необходимо:

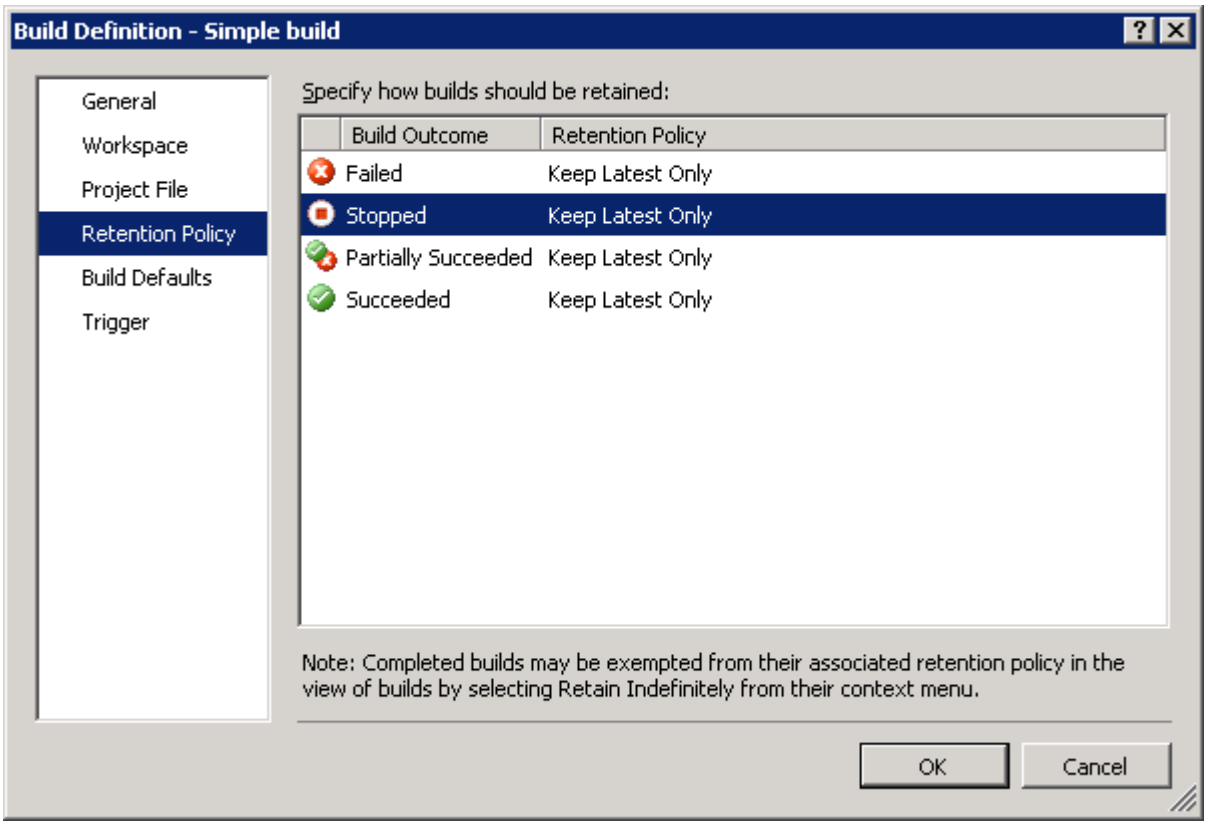
1. Вызвать команду Edit build definition:



2. Задать настройки автоматического запуска на закладке Trigger:

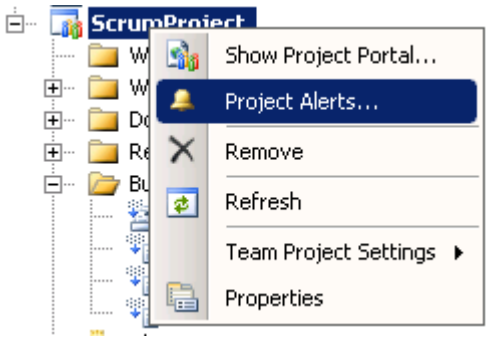


3. Настроить политику очистки сборок на закладке Retention Policy. Это необходимо для того, чтобы избежать быстрого исчезновения места на машине-сборщике и для удаления из базы TFS информации о второстепенных сборках:

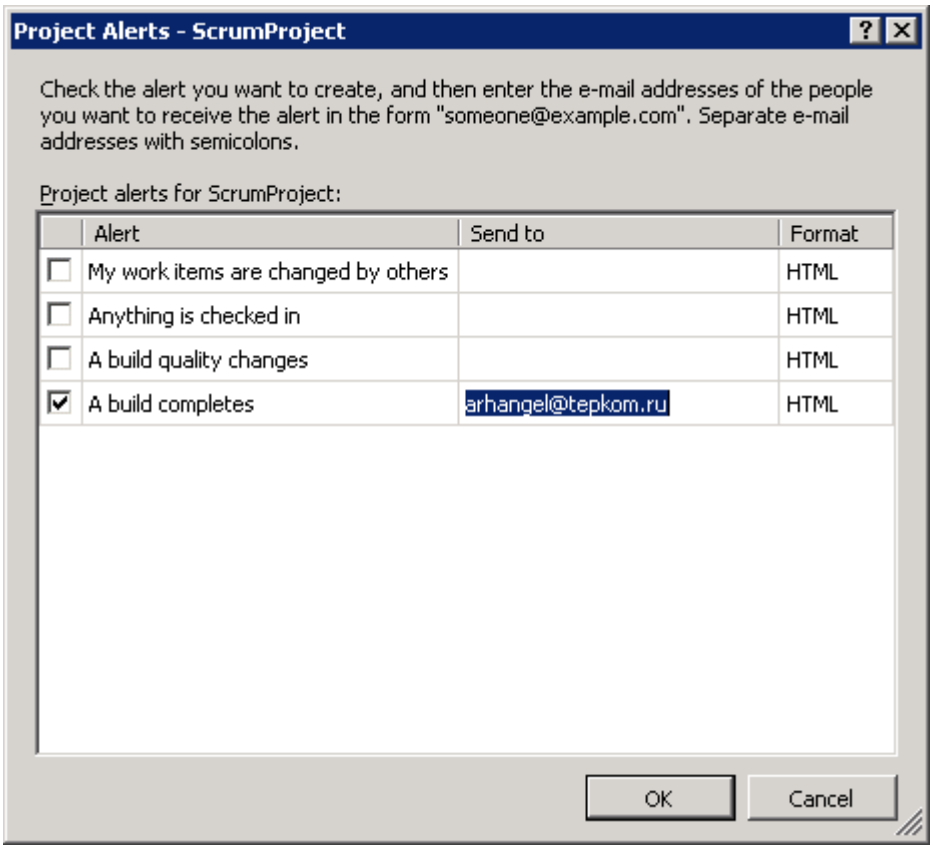


Проведение сборки после внесения изменений наиболее эффективно в том случае, если участники проекта получают нотификации о том, что сборка была проведена. Для того чтобы этого добиться необходимо:

1. В контекстном меню проекта выбрать команду Project Alerts:



2. В списке событий, о которых нужно слать нотификации, выбрать “a build completes” и задать список адресов электронной почты, на которые нужно отправить сообщение:



После настройки простой сборки для запуска при внесении изменения необходимо проверить работу системы – внести некоторое изменение и дождаться сообщения о сборке.

Аналогичным образом можно настроить и автоматический запуск сложной сборки каждый день в определенное время.

Тема 6. Настройка шаблона процесса.

На завершающем этапе, соответствующем окончанию спринта, команды должны провести ретроспективу своей деятельности и сформировать список возможных изменений в процессе и работе с TFS. Все замечания должны быть занесены в TFS как соответствующие элементы работы.

В рамках ретроспективы команда должна предложить некоторые изменения к элементам работы, вовлеченным в процесс, а затем и реализовать эти изменения.

Шаг 1. Ретроспектива

На ретроспективе команда в течение 20-30 минут обсуждает то, как прошел данный спринт, выделяя позитивные и негативные моменты, а также предложения по изменениям.

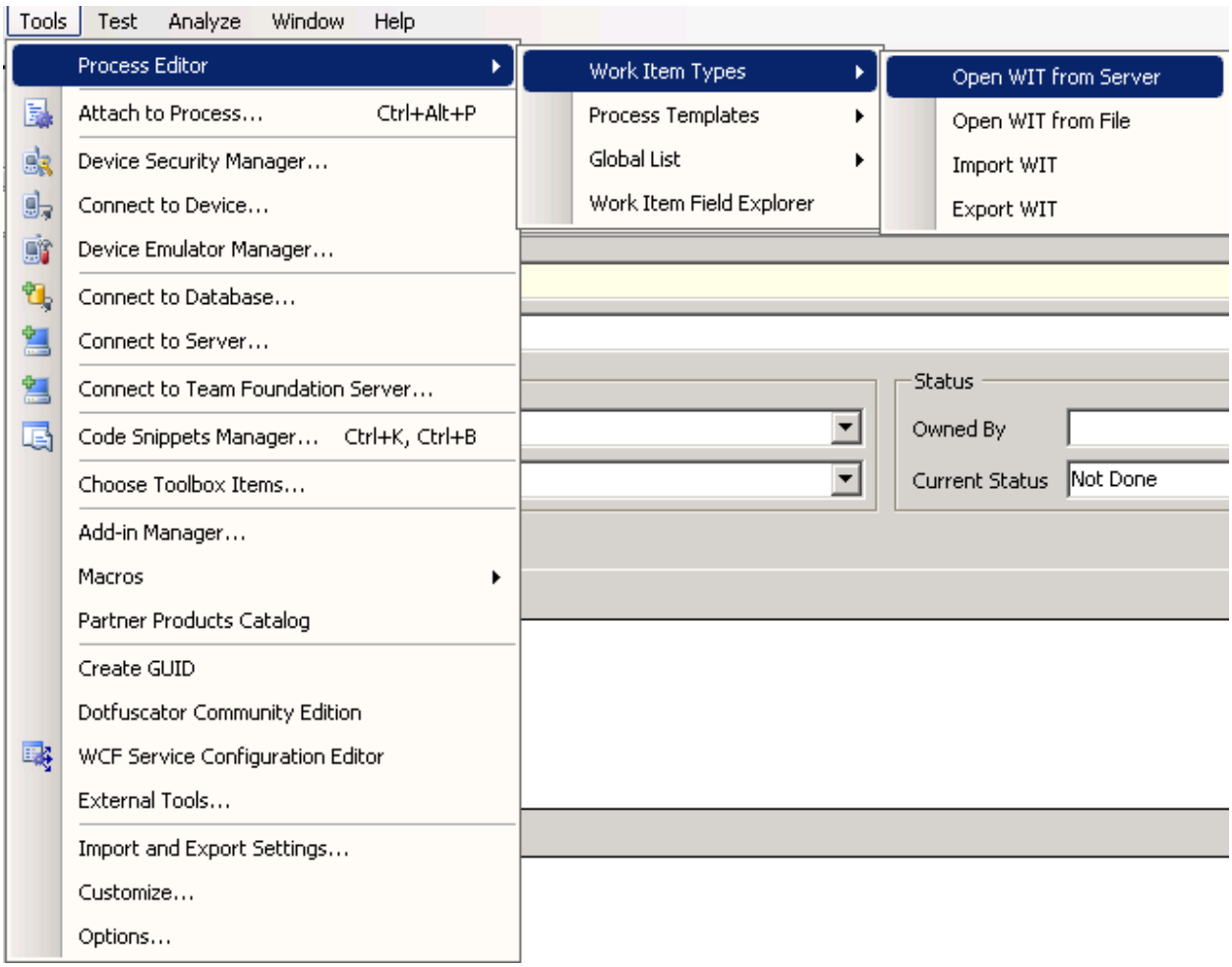
Все комментарии должны быть внесены в соответствующий элемент работы типа Sprint retrospective, а для каждого предложения по улучшению заведены элементы работы типа Sprint backlog item, а для каждого идентифицированного негативного момента, требующего устранения – элемент работы типа Impediment.

Результаты ретроспективы необходимо обсудить с хозяином продукта.

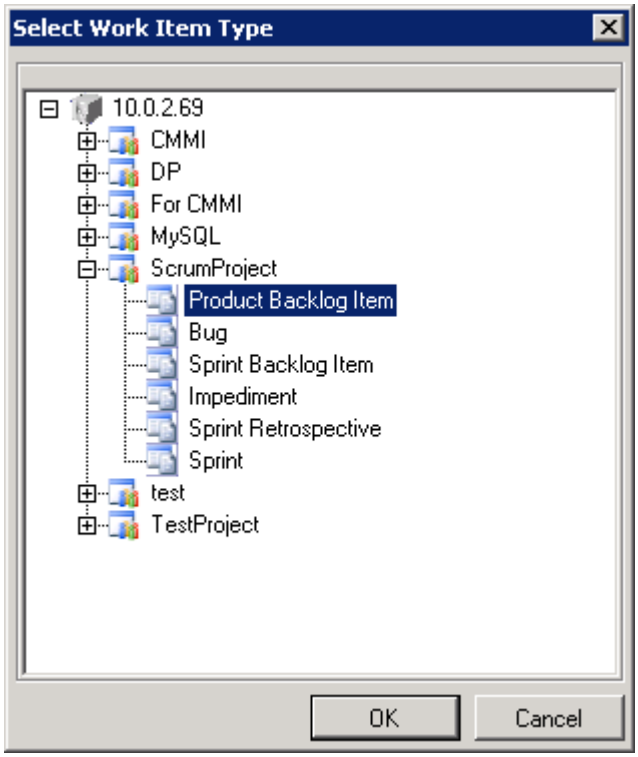
Шаг 2. Изменение элемента работы

На этапе ретроспективы команда должна выявить некоторые изменения в формате и жизненном цикле элементов работы, которые помогут повысить эффективность команды. На следующем шаге им нужно воплотить эти изменения в жизнь. Для этого необходимо:

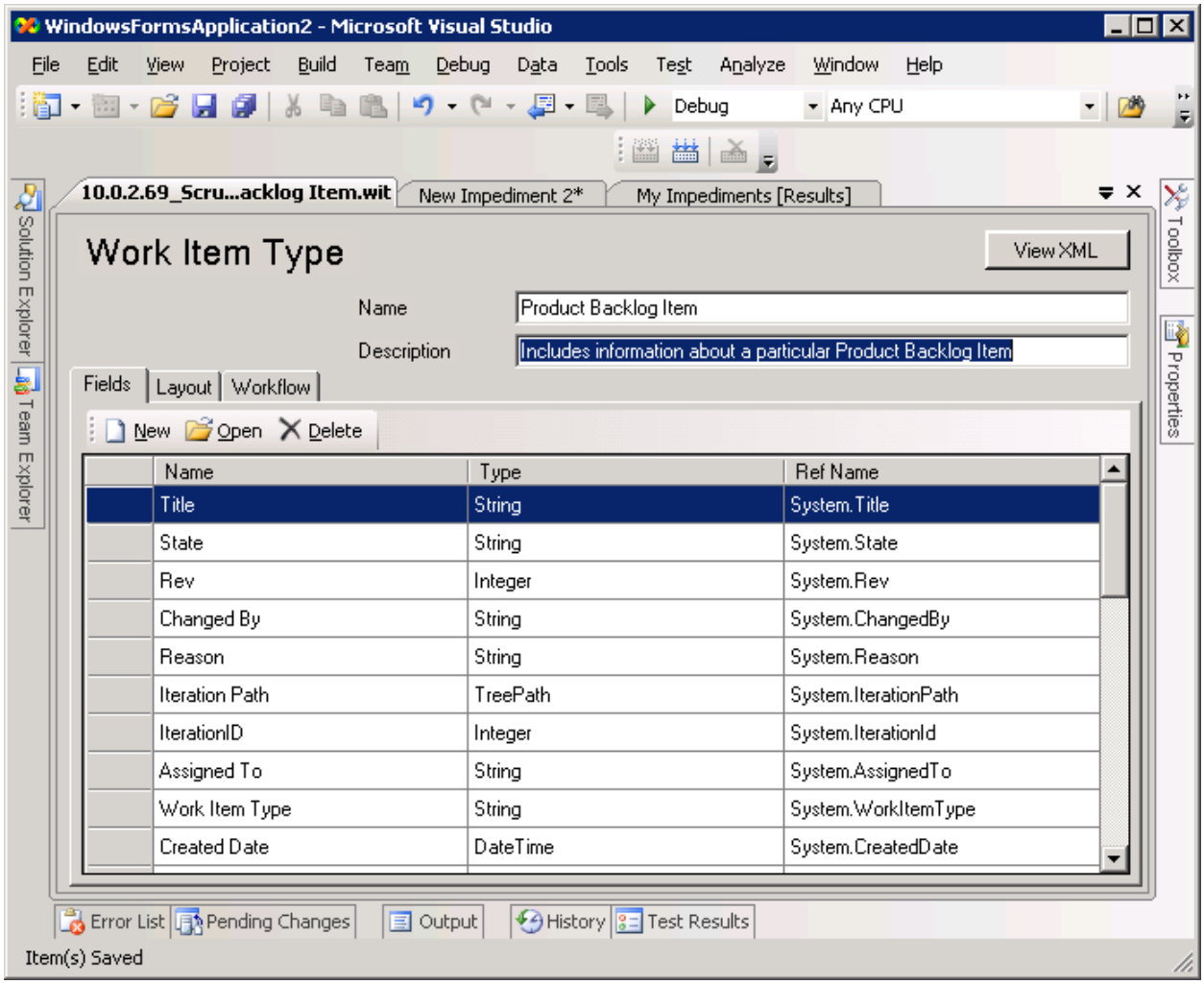
1. Открыть тип элемента работы на редактирование с помощью команды меню Tools:



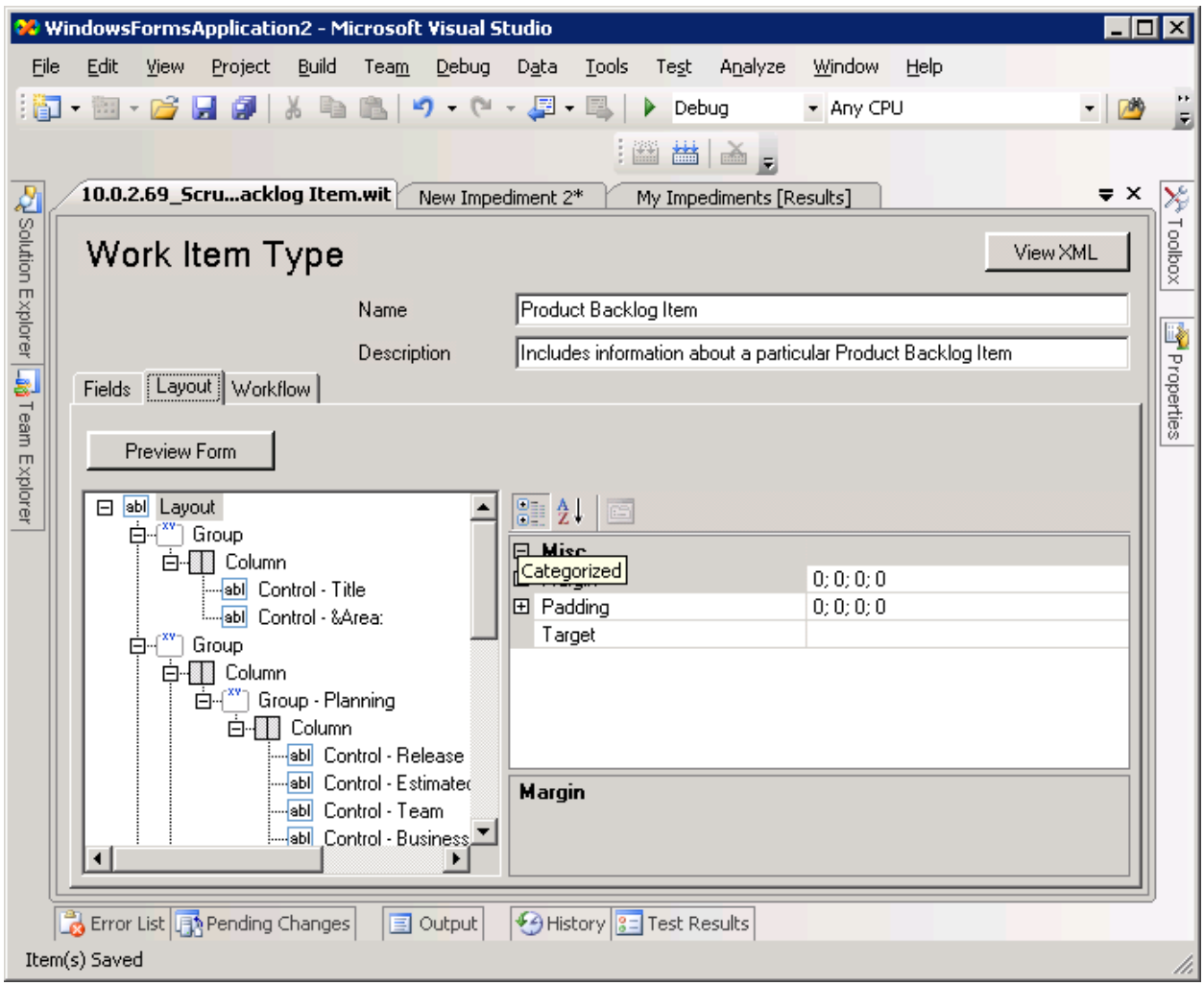
2. Выбрать нужный элемент работы в открывшемся диалоге:



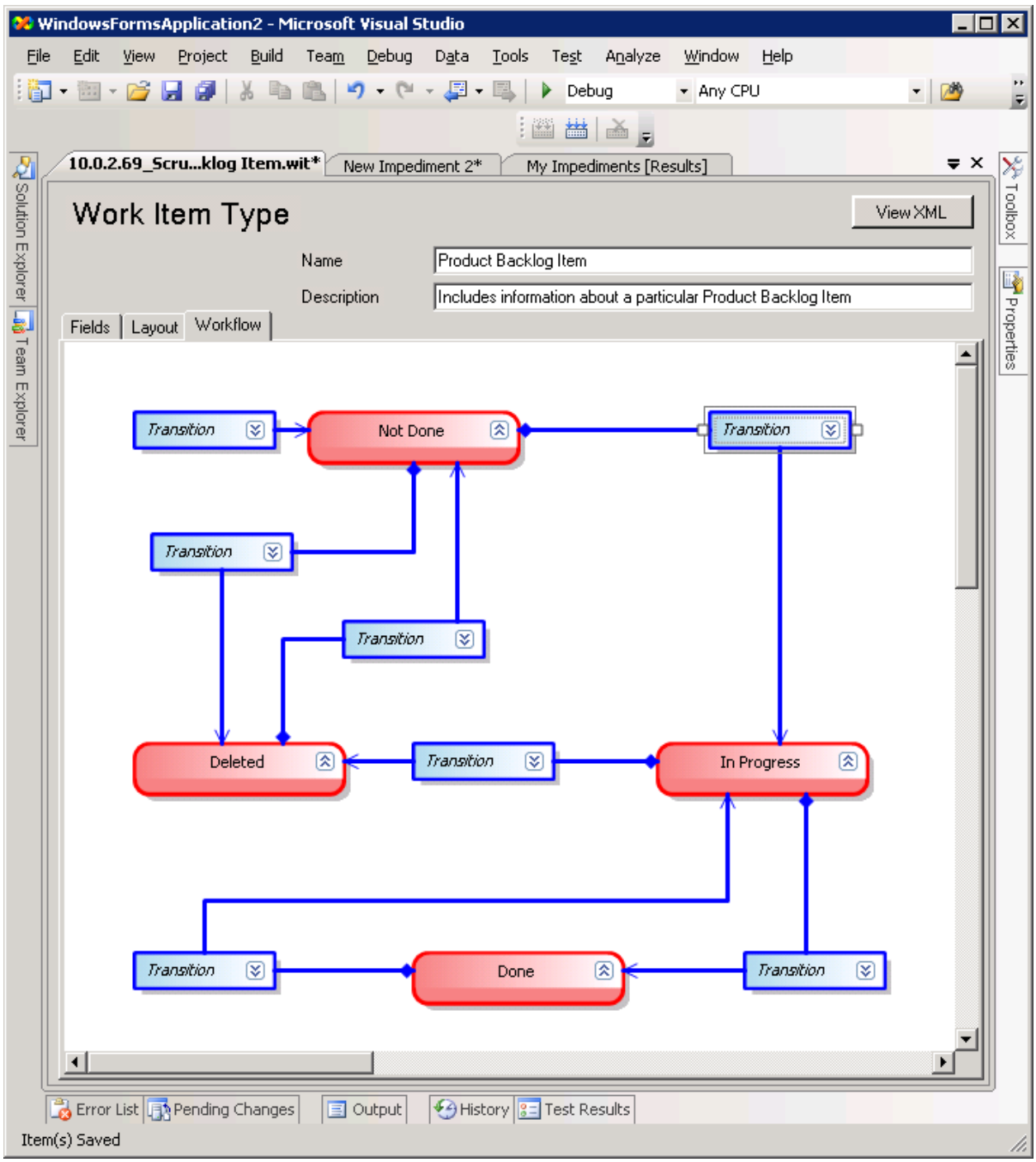
3. На закладке Fields добавить, удалить или изменить необходимые поля:



4. На закладке Layout изменить соответствующим образом визуальное представление элемента работы:



5. На закладке Workflow внести необходимые изменения в жизненный цикл элемента работы:



После внесения всех изменений их нужно сохранить, а затем убедиться, что они применились к существующим и к вновь создаваемым элементам работы.