Requirements Engineering Best Practices

A Guide for Visual Studio and Team Foundation Server Users

A Visual Studio ALM Rangers Project

Table of Contents

Introduction to Requirements Management with Team Foundation Server

Welcome to the Team Foundation Server Requirements Management Guidance! During TechReady7 in July, 2008, practitioners voted for projects they believed were most needed by the Visual Studio ALM Rangers. Requirements Engineering was voted to be one of the top 2 needs. This document will address the People, Process, and Technology guidance for Requirements Engineering (RE) using Team Foundation Server.

The goal of this guidance is to provide formalized Microsoft field experience in the form of recommended procedures and processes, Visual Studio System and Team Foundation Server configurations, and skill development references for the Requirements Engineering discipline of your application lifecycle. Due to the variability and breadth of methodologies used throughout the industry, this guidance takes on this objective in three (3) ways.

- **Generic Requirements Engineering Guidance** – There are RE practices that are relevant regardless of the methodology used. For example, the natural progression from business level abstraction to functional capabilities to qualities of service are defined in some way whether the development organization is performing traditional waterfall according to Institute of Electrical and Electronics Engineers (IEEE) standards or an Extreme Programming implementation of Agile Methods. This guidance is given at a conceptual level using specific Visual Studio/Team Foundation Server examples when applicable.
- **Traditional Development** – Guidance will be offered that aligns mostly with IEEE standards for requirements engineering. As such, specific guidance for business requirements elicitation, specification, and validation leading toward functional requirements specifications and technical requirements specifications will be described.
- **Agile Development** – Even within Agile Methods there are several methodologies that accentuate or highlight different practices within the movement. As such, we are focusing mainly on Scrum as the chosen methodology and will make reference to other methods when appropriate.

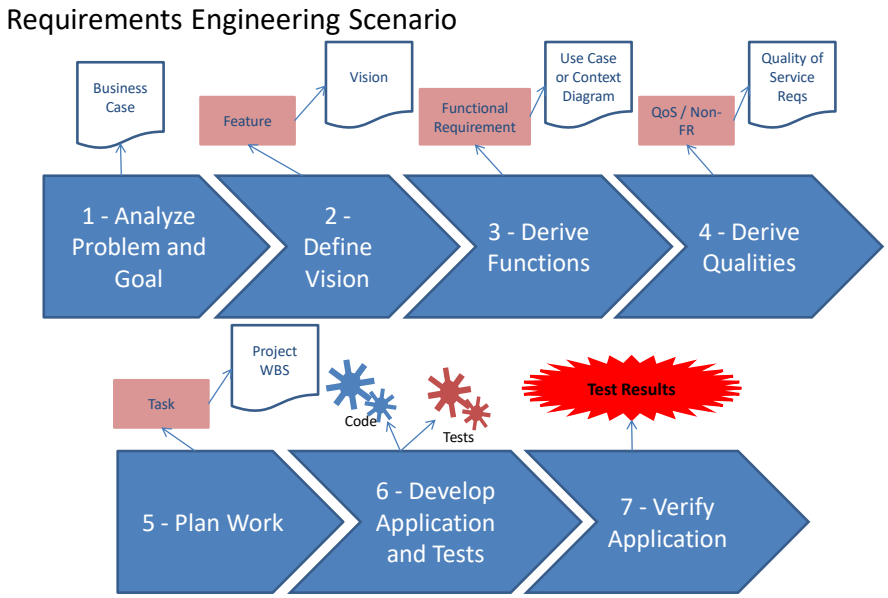
Requirements Engineering Scenario

Requirements engineering, whether performed for a project executing against a waterfall lifecycle or an agile lifecycle, will need to take into account many of the same core pieces of any project. The following list describes the elements that need to be elicited, described, analyzed, validated, linked, evolved, and tested through the course of the project:

- Business Requirements and their Acceptance Tests
- Functional Requirements and their Scenario Tests
- Technical Requirements and their non-functional and quality of service thresholds
- Design Specifications and their Application Architecture Component Tests

- Source Code and its Unit Tests.
- User's documentation

This guidance describes this list in much more detail in the Requirements Management Planning and Traceability sections. For now, we will use this list as the basis to discuss a requirements engineering scenario. This scenario will walk through the beginning elicitation of the business level requirements all the way to executing tests against a build. It will describe the goals of each step of the process and how Team Foundation Server can be used to support each step.



This scenario will serve the basis for discussion throughout this guidance and will be referenced as we describe alternatives based on Generic, Traditional, and Agile practices seen throughout Microsoft and the Industry as a whole. In each scenario step described below, we point out where in the guidance more information can be found to understand how Visual Studio and Team Foundation Server can provide additional support.

1. **Analyze the Problem and Goals and Define Vision -** Business Exec sponsors a project to solve some business problems or achieve business goal.
   The Business Analyst will elicit these as product requirements in the form of a feature set or business requirements. We can store these in Team Foundation Server as a "Feature" work item.
   a. Initial elicitation will be targeted at the Business Executives, but as product features are derived, business domain experts will contribute to the feature set. **(RM Planning & RM Elicitation Topic Areas)**

b. Validation at this level will ensure that the features described are not only accurate, but contribute to solving the business problem or achieving the business goal. **(Requirements Validation Topic Area)**
c. Problems and Goals are usually captured in the form of a business case document where, in normal IT organization sponsored by a CIO, the project manager gathers preliminary estimates of the cost of solving the problem through an app-dev project and contributes to the cost-benefits analysis. This type of activity is out of the scope of requirements engineering in this guidance. The analysis of the business case and further elicitation of the sponsors and domain experts will result in business requirements or product features that will be specified in documents for the project; usually the Business Requirements Specification/Document (BRD). **(RM Planning & RM Specification Topic Areas)**

2. **Derive Functions -** By performing Analysis of the features, the analyst will derive a functional context model or use case diagram for all of the functional scenarios that realize the features. We can store these as User Story work items (MSF for Agile) or Requirements categorized as "Functional" type (MSF for CMMI) work item and link them to each feature for which they realize.
    a. Plan for Requirements – **(described in RM Planning Topic Area)**
    b. The analyst will analyze the features and prepare for eliciting functional requirements from user stakeholders and business subject matter experts. Their analysis will guide them through eliciting the correct functional requirements, documenting them, and validating that the requirements are accurate. **(RM Elicitation, RM Analysis, & RM Validation Topic Areas)**
    c. Link the derived functional requirements to the feature requirements – **(RM Traceability Topic Area)**
3. **Derive Qualities of Service and Non-Functional Requirements -** By performing more Analysis of the features and the functional requirements, the analyst, working with the other product constituents (Infrastructure, Operations, Architecture, Database Administrators, Usability Lab, Subject Matter Experts, Security, Governance and Compliance, etc…) they will identify non-functional requirements and Qualities of Service. Again, we can store them as work items in Team Foundation Server and link them to each feature or function for which they are derived.
    a. Analysis of Functional Requirements to derive Quality of Service and Non-Functional Requirements - **(RM Analysis, RM Elicitation, & RM Specification Topic Areas)**
    b. Validation that the Qualities and Non-Functional aspects are accurate **(RM Validation Topic Area)**
    c. Link the derived requirements to the functional requirements – **(RM Traceability Topic Area)**
4. **Plan Work -** The project manager will work with the development team to identify development tasks (whether it's agile or waterfall or anything in between, tasks are still identified and estimated in some way) for the functional and non-functional requirements. We can store these as "Task" work items and link them to the functional or non-functional requirements for which they were planned.
    a. Plan the development and Testing Effort – **(RM Analysis Topic Area)**

5. **Develop Application and Tests -** The development team can develop application code projects and unit test harness projects (performance tests as well) and check them into source code. At check-in time, they link to a task, and we can even force them to do it.
    a. Link Change Sets to Tasks and Requirements – **(RM Traceability Topic Area)**
6. **Verify Application Quality -** We execute the unit tests and we can demonstrate with a report which tests passed and failed, which builds they were linked to, which change sets went into that build, and which work items were linked to those checked-in change sets. All wonderful and we can do that with Team Foundation Server. We execute functional tests with Manual harnesses and Web Test harnesses as well as Generic test harnesses that wrap external functional tests (i.e. Quick Test Professional) and publish the results. Reports can demonstrate that work items are covered by all types of tests through the Team Foundation Server Data warehouse.
    a. Run tests, publish results, generate coverage reports – **(RM Traceability Topic Area)**
7. **Test Planning** – Why is test planning included as a requirements engineering topic? Well, test planning ensures that validation has a placeholder for verification. Validation provides the checks to ensure that the project team understands the goals of a requirement and can gain agreement with the requirement's source; a business sponsor or stakeholder. Verification provides the test that the team has delivered what they've agreed to deliver. Test Planning provides the place holder and traceability for all of the test management work. So,… After step (1) above, but before step (2), we now have a solid mechanism to plan the acceptance tests for the feature requirements. In fact, the test planning effort at each level can be accomplished with the Visual Studio 2010 Test and Lab Manager edition. So, we can create a series of test plans and suites in Test and Lab Manager that provides for developing a robust test plan report that displays which tests have not been planned, which were planned but not implemented, which were implemented but not executed, which were executed but failed, and which executed and passed. The data is all there, and we now have a way to report on it and sort it by the work item for which they belong.
    a. Plan tests for system requirements – **(RM Traceability, Validation, and Specification Topic Areas)**

Each of the steps above can be supported with "out-of-box" features in Team Foundation Server. Some, though, are not yet covered by features of Team Foundation Server. For these circumstances, our guidance offers work-around or manual solutions that provide a means by which our users can achieve a comprehensive solution with Team Foundation Server anyway.

> **Note:** The guidance for requirements analysis found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Disclaimer

In the development lead's 20 year career practicing, implementing best practices, teaching, and coaching application lifecycle management capabilities for his customers, it has been found that

Requirements Engineering tends to be the most opinionated discipline within the software and systems development community. As such, this guidance has been developed with an approach that begs criticism and change management. A team of more than 20 practitioners throughout Microsoft Consulting Services, Product Management, and the MVP program have come together to debate and define a guidance offering that takes into account industry best practices and Microsoft specific best practices in the Requirements Engineering discipline.

Visual Studio ALM Rangers

This content was created in a Visual Studio ALM Ranger project. Visual Studio ALM Rangers is a special group with members from the Visual Studio Product group and Microsoft Services. Their mission is to provide out of band solutions to missing features or guidance.

This guide targets the Microsoft "200-300 level" users of Team Foundation Server. The target group is considered as intermediate to advanced users of Team Foundation Server and has in-depth understanding of the product features in a real-world environment. Parts of this guide may be useful to the Team Foundation Server novices and experts but users at these skill levels are not the focus of this content.

Before executing on your requirements management plan, consideration should be given to the methodology being used as well as any governance or legal compliance requirements the organization will need to maintain. In addition, adoption of any organizational change as a result of a requirements engineering process implementation should be considered as practitioners have a tendency to resist large changes that alter the way they do their job. These type of changes can impact productivity in organizations that impose process changes without a plan for their adoption.

Our hope is that this guide covers the great 80% segment of requirements engineering needs. In addition to this guide you will find separate methodology scenarios, Q&A, tutorials and updated traceability and documentation strategies. We encourage ongoing contributions from our user community to submit ideas for new Scenarios that we will add to these packages. These new packages will become available on CodePlex or MSDN and will be refreshed based on user feedback.

Vocabulary

This guide uses vocabulary and general nomenclature that are used in the application development community. Here is a list of some of the commonly used terms, many of which are used in this guidance. Language may vary in your organization so some translation may be required.

Term	Definition
Acceptance Testing	Testing that brings the customer into the final validation process in order to gain assurance that the "product works the way the customer requires." Acceptance tests are typically based on specific usage scenarios that execute in the user environment.
Analyzing Requirements	The process of examining the requirements for correctness,

	vagueness, feasibility, and other quality characteristics, then decomposing into more detailed requirements.
Attributes	Quality characteristics of the product (e.g., reliability, usability, etc.)
Availability	Degree to which system is operational and accessible when required for use.
Baseline	Specifications or other work products that have been formally reviewed and agreed upon, thereafter serving as the basis for further development. Contents of the baseline can be changed only through formal change control procedures.
Bidirectional Traceability	An association among two or more logical entities that is discernable in either direction (i.e., to and from an entity). (See also Tracing Requirements).
Business Rules	The rules and constraints governing a particular business process or workflow.
Change Control	An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.
Change Control Board	A group of people responsible for evaluation of a change, approving or disapproving proposed changes to configuration items, and ensuring implementation of approved changes.
Context Free Questions	Questions that force the interviewer to listen before attempting to invent or to describe a potential solution. Questions that focus on first gaining a real understanding of the problem and what solutions have already been envisioned. Example context-free questions include: Who is the user? Who is the customer? Are their needs different? Where else can a solution to this problem be found?
Correctness	Attribute: conforms to standards and specifications.
Defective Requirement	A requirement that is incorrect, ambiguous, unverifiable, untraceable, incomplete, and/or inconsistent.
Efficiency	Attribute: relative use of resources (memory, time, network bandwidth).
Flexibility	Attribute: can change mission, function, data to satisfy other requirements.
Focus Group Technique	A requirements gathering technique that follows a script and an organized set of interview questions with a group of people. The group consists of 3 – 10 users, experts, or customers, several developers, or marketers.
Functional Requirement	Capability required of a product in terms of function or service
Hardware Requirement	A requirement that is intended to be implemented in or by hardware.
Integrity	Attribute : (of use) performs without failure due to unauthorized access; (of data) preserves content and order of data and work products.
Interaction Diagrams	A requirements analysis technique used to illustrate the interactions between the various entities in the business rule, storyboard, scenario, or use case.

Interoperability	Attribute: can couple software of one system to another
Interview	One-on-one session with a user in the workplace
Joint Application Design (JAD) Session	A series of joint meetings of users and technical staff for the purpose of specifying requirements. Generally follows a structured process of information gathering and review, a facilitated session with defined deliverables and activities, and documented results.
Maintainability	Attribute: ease of locating and fixing a failure within a time period(during use).
Meta-questions	A question about questions. Typically used by the interviewer to determine if the interviewee understands the intent of the interview. Example meta-questions include: Do my questions seem relevant? Are my questions clear to you? Are your answers official?
Non-Technical Constraints	Requirements related to product, delivery, and production (e.g., cost and schedule)
Packaged Solution	An off-the-shelf solution that usually supports a number of business needs, functions, etc. Normally the source code is not available.
Participatory Design	A requirements gathering technique originating in Scandinavia. The software is seen as part of a functioning, complex organization. The user is understood in terms of his/her place in the entire system. An examination is made of the various roles and parts of the system and a determination is made as to which should be done by software and which by humans. There is active participation by the users in designing the solution.
Portability	Attribute: can transport for use in another environment.
Prioritizing Requirements	A technique for ranking the importance of the requirements from most to least important.
Prototype	A partial implementation of a software system, built to help developers, users, and customers better understand the requirements of the system.
Purchaser	One who acquires the product for the user
Reliability	Attribute: performs without failures within specified time period
Requirement	• A condition or capability needed by a user to solve a problem or achieve an objective • A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents(IEEE definition)
Requirements Baseline	The repository of the agreed to and approved system requirements, software requirements, hardware requirements, operational requirements, training requirements, use-case models, etc.
Requirements Engineering	The disciplined application of principles, methods, and tools to describe proposed systems behavior and constraints throughout the product/project life cycle.
Requirements Gathering	The process of eliciting, organizing, analyzing, and validating requirements prior to incorporating them into the requirements baseline.
Requirements Management	The establishment and maintenance of a requirements baseline for

	software engineering and management use. Software plans, products, and activities are kept consistent with the requirements baseline.
Requirements Modeling	The activities that transform a set of requirements into a graphical and/or textual model representation of the system as it will operate if the requirements are implemented.
Requirements Specification	Documentation that defines the product to be built and describes its external behavior.
Reusability	Attribute: can be converted for use in another application.
Scenarios	An outline or synopsis of a business rule, storyboard, or use case that puts the intended functionality into the context where it will operate.
Software Development Lifecycle	The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase.
Software Product Engineering	The group of activities required to consistently perform a welldefined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
Software Requirement	Technical response to user requirements, often documented in a software requirements specification; drives design of the product.
Sponsor	Person who provides funding for the project.
Stakeholder	Person who will be affected by the product, and has a direct or indirect influence on product requirements.
Storyboard	Graphical representation of the flow through the system. The flow can be the main flow or it can be a special instance that is critical to get correct.
System Interface Requirement	A requirement that describes the data communications, processing, power, or other interface between subsystems.
System Requirement	A condition or capability that must be met or possessed by a system or system component to satisfy a condition or capability needed by a user to solve a problem.
Technical Constraints	Characteristics of the product environment that affect implementation and/or design of the product.
Technical Reviews	A review by subject matter experts of the correctness of the technical content of the work product under review.
Technical Requirements	Capabilities of the product in terms that drive development of the product
Testable Requirement	A requirement for which an unambiguous test case can be written.
Testability	Attribute: can verify (test) specified operation and performance
Traceable Requirement	A requirement with a unique identifier or label. A requirement is traceable, if and only if the origin of each of its component requirements is clear, and there is a mechanism that makes it feasible to refer to that requirement in future development efforts.

Tracing Requirements	The process of tracing backward from the current requirement to the previous stages of definition or development and forward from the current requirement to all of its subordinate requirements (plans, design documents, flowcharts, code, test cases, etc.). (See bidirectional traceability.)
Unambiguous Requirement	A requirement is unambiguous if and only if can be subject to only one interpretation.
Unified Modeling Language(UML)	A graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system. UML provides a set of modeling elements, notations, relationships, and rules for use that could be applied to a software development activity.
Usability	Attribute: Effort required for the user to be able to use the product, for example, time required for training or learning new operation
Use Case	A use case describes a sequence of actions a system performs that yields a result of value to a particular actor.
User	One who will use the product.
User Constituency	Subset of users, categorized by product characteristics or ways of using the product.
User group	Formal organization of users which convene periodically to discuss product use and improvement ideas.
User Representative	One who represents users to the product developers.
User Requirement	A high level system requirement from the user. User requirements should not specify implementation.
Validating Requirements	Validation focuses on whether the product works as it is supposed to do. Requirements validation involves inspecting the relationships between the tests (and test results) and the system being tested. Does the as-built system satisfy the requirements?
Verifiable Requirement	A requirement that can be verified through analysis, demonstration, inspection, or test.
Work Product	Documentation, software, training or any intermediate product created in the process of developing a product; may be for internal use only or for delivery to the customer.

Requirements Management Planning

Many organizations begin development with a loosely conceived idea of what they want to build and focus on getting to the design. Formal definition of how to capture, specify, and manage the evolution of requirements is often left undefined. Because of this, projects often do not allocate enough time for the elicitation and validation effort required to get the project's requirements correct. As a result, the team gets to the end of the project and run out of time or delivers defects that can be attributed to a misunderstanding of requirements.

A Requirements Management Plan should be developed to specify the information and control mechanisms which will be collected and used for measuring, reporting, and controlling changes to the product requirements.

The plan should be saved in the project's process guidance folder of its documentation library, along with the rest of the project's process guidance.

- Identify Methodology (we will describe scenarios related to Agile processes with a focus on Scrum as well as traditional methods using waterfall)

  The identified methodology should be more of an implementation, or development case, specific to each project that executes according to the company's methodology.

- Identify Traceability Items

Generic Methodology-Free Elements

In following guidance by the Capability Maturity Model Integrated (CMMI), International Association of Electrical and Electronics Engineers (IEEE), International Standards Organization (ISO), or other measurement or standards bodies, a natural evolution of business to functional to technical requirements should be established and maintained using bi-directional traceability. In effect, this means that the organization should identify requirements types that allow for the gathering and storage of the requirements at multiple levels to depict the growth of the understanding of the requirements through their lifecycle. This applies to all software development, whether performing traditional waterfall development or extreme agile (XP).

Project requirements begin as a business goal or problem to which a solution will achieve success. That solution is identified through the analysis of the business goal or problem within the boundaries of the company's business domain.

Business goals and problems are analyzed to determine the needs for a solution that are then analyzed to determine features of that solution. Once identified, further analysis of the features along with prioritization based on buy vs. build vs. hybrid (buy and customize) along with constraints defined by politics, funding, technology feasibility, enterprise architectural guidance, etc… will allow the team to select those features of a solution that determine the scope of a project within a portfolio. It is at this point within a company's portfolio that the requirements will be transformed into the funding for a

project that will achieve the original business goal or solve the identified business problem. In an IT organization for a financial services company for example, that means a funded project in the portfolio. For a shrink-wrap software vendor, that means a set of features for a product that will be released to market.

Documenting the Plan

The results of each of the sub-topics described below should be documented in a "Requirements Management Strategy" document. This document describes the team's work items and traceability strategy with all of the attributes, templates, checklists, and reports that offer usage of the strategy. This document should be stored in the process guidance library in a "Requirements" folder.

Traceability

Once a project begins, the features of a solution are analyzed into functional scenarios that can be assigned, estimated, further analyzed for qualities of service and evolved through the cycle of software development. ***Figure 1*** represents a generic hierarchy of the evolution of Business requirements through to delivered source elements and their tests with a potential representation in a vision document. Most software development projects are allocated funding after a business case is approved and a management body is assigned. At this point, business problems and solution needs and alternatives have been vetted and an initial set of product features is established. This is the "sweet spot" from which to establish a link to or starting point in Team Foundation Server. This is depicted in the traceability graphic below.

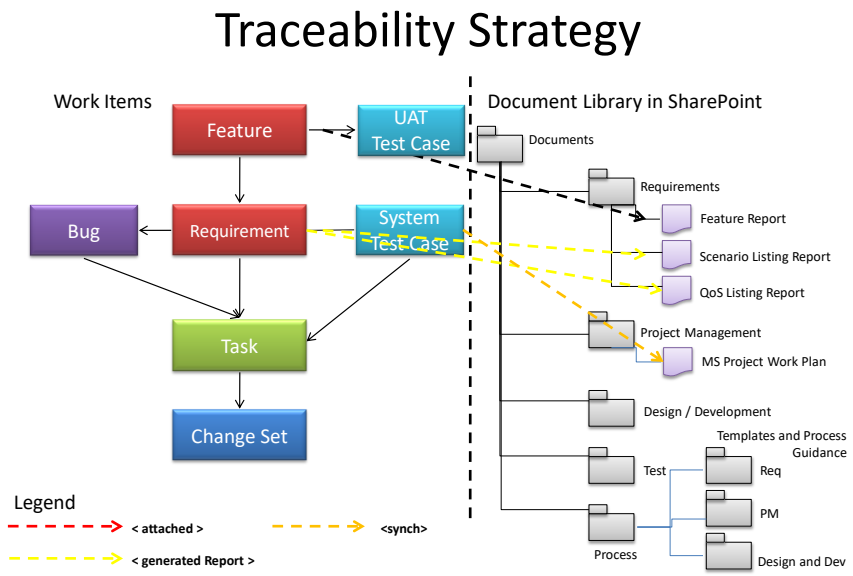


Figure 1: Generic Trace Strategy

The accompanying topic called Requirements Traceability will describe Team Foundation Server usage and support for traceability in much more depth than described here.

Roles and Responsibilities

During the course of defining the requirements traceability strategy, responsibilities must be aligned with the various levels of evolution. For example, a CEO or business representatives will own the responsibility for defining the company's or software vendor's business goals or problems that need to be solved, while a test analyst or test manager will be responsible for identifying the testable acceptance criteria for the functional requirements of the system. In the above generic trace hierarchy, we recommend roles equivalent to the ones identified in the following responsibility table for each trace level:

Requirement or Evolutionary Level	Responsible Role
Business Goal	CEO or business representative
Business Problem	CEO or business representative
Need	Business Stakeholders (at times includes CEO and business representatives, but could also include users of the ultimate application or system)
Feature	Business Analyst(s)
Functional Requirement (scenario based)	Business Analyst
Quality of Service	Business Analyst, Enterprise Architect, Application Architect, Infrastructure Architect, DBA, Test Engineer, User Experience Engineer, etc… Determined by the sub-type of the quality of service.
Test Case	Test Engineer for functional requirements, Performance Test Engineer for scalability, load, and reliability requirements, Developer for Source Units and Components, and others depending on test types planned.
Source Code	Developers and Application Architects

Each role should be described in terms of the activities they perform to achieve the deliverable or artifact and the acceptance criteria for the artifact.

Requirements Attributes

The next element of the generic plan is to identify the measures and attributes that will provide prioritization assistance and visibility into project progress. For example, the following attributes will be helpful:

- Identification – some numbering mechanism that allows for easy retrieval and isolation from the rest of the set of requirements.
- Status – New, Assigned, Ready for Test, Test Passed, Complete.
- Risk – Factor of Probability and Impact
- Size – Rough order of Magnitude (Small, Medium, Large, Infinite)
- Estimate – Work Effort required to implement and successfully pass all tests

- Architectural Impact – Complexity in terms of function points or depth across the application architecture
- Business Area – Taxonomy referring to the business unit that owns the usage.
- Assignment – Practitioner responsible for delivery of the requirement.

It will be important to identify the attributes that are applicable to each requirement type separate from the others. For example, scenario requirement might have attributes specific to the steps of the scenario while a business requirement will have attributes specific to return on investment or other business case data.

The previous data identifies the structure to support requirements engineering. The rest of the plan should specify the dynamic element of evolving requirements through a software development project.

Reporting

In order for a trace hierarchy or requirements strategy to be effective, the development team and project management need a mechanism to understand the progress of the requirements through the life of the project as well as to identify impacts of changes. Reports of the project data specific to the requirements achieve this goal. The following are examples of reports that assist in managing the requirements through the project:

- Feature Completion Summary – This report is a listing of each feature with its scenarios nested below. The scenarios can provide their detail about status and work remaining to provide an understanding of Feature Completion.
- Scenario Test Coverage – This report is a listing of each scenario with its tests nested below. The report will help understand testing coverage (or lack thereof) and the success or failure of the results of the test execution.
- Vision/Scope – This report provides the detailed listing of features with their attributes as they align with business goals and objectives.
- Scenario Listing – This report provides a listing of the functional requirements with their attributes. Sorting will allow the team to specify priorities or completion status.
- Scenario Completion Status – This is a listing of the Scenarios (or functional requirements) with their traced tasks nested within their descriptions to provide comprehensive planning and completion status for project managers or Scrum Masters.

The Traceability guidance describes and demonstrates these and other reports necessary for complete requirements traceability monitoring.

Tools

Using Visual Studio and Team Foundation Server for requirements capture, storage, and tracking is a given. This document's goal is to provide requirements engineering guidance specifically in the context of Visual Studio/Team Foundation Server. This section of the requirements management plan, however, should articulate the details of tools used in concert with Visual Studio/Team Foundation Server for elicitation of the requirements in the trace hierarchy, linkage to 3rd party vendor products, as well as

Visual Studio/Team Foundation Server specific usage for each requirement type and capability within the trace hierarchy.

Examples of tools used through the lifecycle might be:

- IRise – A story boarding graphic design tool to specify the details of business scenarios. Artifacts from IRise should be stored or linked to specific work items in Team Foundation Server.
- Team Foundation Server Work Items – Feature work items should be defined, User Story work items should be defined and traced as a result of the feature analysis using the link capability in Visual Studio, Task work items should be defined that link to scenarios as a result of iteration or project planning, and version control change sets and documents in the Windows Sharepoint Services (WSS) portal should be linked to tasks.
- Templates and Documentation Work Products – Sharepoint should be organized to make it easy for a team member to retrieve a template for specific work as well as a storage location for the resulting work product of the template.

Change Management

Changes to requirements can, and often time do, result in increases in project scope that lead to delivering over budget and schedule with low quality. Because of this, a rigorous policy for managing changes to requirements should be documented in the project's requirements management plan.

Specifically to this, the following concerns should be addressed by the requirements change management policy:

- Change Request Processing and Approval – this describes the process by which requirements changes should be submitted, reviewed, and provisioned. This should include the process for negotiating requirements changes with the customer, which will vary depending on traditional approaches or agile approaches, and any contractual processes, activities, and constraints.
- Change Control Authority – Describe who is authorized to approve requests for changes. Often times this is a formal group called a Change Control Board, but, in the case of pure agile projects, it can be provided by the customer or product owner working with the Scrum Master. This section of the plan should describe procedures for processing change requests and approvals to be followed by the change authority.
- Change Control Mechanism – In this guidance we specifically recommend using a work item to store change requests for requirements. It can be a bug, or a new work item specific to requirements. Either way, the work item provides the technical implementation of the work to be performed by the change management process (attributes, workflow, assignments, etc…)
- Baselines – A description of mechanisms for capturing complete sets of requirements as a baseline for defined methodology milestones should be described. This description will describe the procedures and mechanisms for capturing new baselines based on changes and a reporting mechanism to compare the baselines from one milestone to the next. This mechanism often times provides the technical implementation that can support compliance to the Food and Drug

Administration's (FDA) regulation for digital signatures on requirements and changes (CFR-21, Part 11).

Workflow and Activities

Each requirement type in the trace hierarchy should have a work flow described. This workflow can be implemented using the work flow structure on each of the work items used for the requirement types.

In addition to individual work item work flow, the requirements process overall should be described in terms of evolution of requirements sets and milestones that align with the project implementation. In terms of traditional methodologies, that means describing the milestones, responsibilities, and approvals for the business requirements, functional requirements, and technical requirements. For agile teams, that means incorporating requirements reviews for iteration planning and/or iteration retrospectives.

Planning the tasks for Requirements Elicitation and Gathering

In order to establish the planning tasks for eliciting requirements, identify the many sources and profiles of them for the typical software application. It is essential that all potential sources be identified and evaluated for their impact on the application or changes to it. The Elicitation topic covers this in much more depth, but the typical requirements sources are:

- Goals
- Existing Requirements
- Stakeholders
- The operational environment
- Domain knowledge
- The organizational environment

Once the requirements sources have been identified, you can start to plan eliciting requirements and define tasks from the plan. The planning tasks for elicitation must include activities for gathering information from users, customers or other stakeholders based on elicitation techniques. Those techniques are:

- Scenarios
- Interviews
- Prototypes
- Facilitated meetings
- Observation
- And others

Again, the Elicitation topic area covers this in much more depth.

In addition, Team Foundation Server supports the elicitation planning activities and integrates well with Microsoft Project to support complete work breakdown planning and execution.

Scrum / Agile Elements

This section offers specific guidance for developing a requirements management plan for agile projects; more specifically, Scrum projects. Though agile purists will describe this as an unnecessary and overly burdensome activity, the author of this guidance offers this up for the "Regular Schmoe" that wants to learn by example and hand-holding. This section will focus on the requirements types and traceability strategy to support generic agile with additional specificity on Scrum as it is one of the most popular agile methods in use today.

Documenting the Plan

As stated earlier, agile project teams do not advocate writing documents for anything more than giving assistance with getting the job done. In this case, minimal is more. For an agile project establishing a requirements management and tracking policy, it is still advised to write a simple document that describes the teams work items and traceability strategy and a simple description of how to use it. This document should be stored in the process guidance library in a "Requirements" folder.

Scrum Traceability

Traceability in a Scrum project begins with the product backlog. The product backlog is comprised of scenario requirements, defects (In a pure agile world they don't 'hang-over' from a previous sprint, but this document takes into account the extremely large world of "Scrum-Butt" practitioners), and change requests.

Product ownership is a discipline unto itself and is not addressed in this guidance. The product owner will negotiate their own methodology to build the backlog and Team Foundation Server is its storage location. Using the traceability diagram drawn above, a scrum project uses the Functional Requirements, Quality of Service Requirements and Tasks as the work items of substance in the hierarchy. Refer to Figure 2 below for a graphic representation.

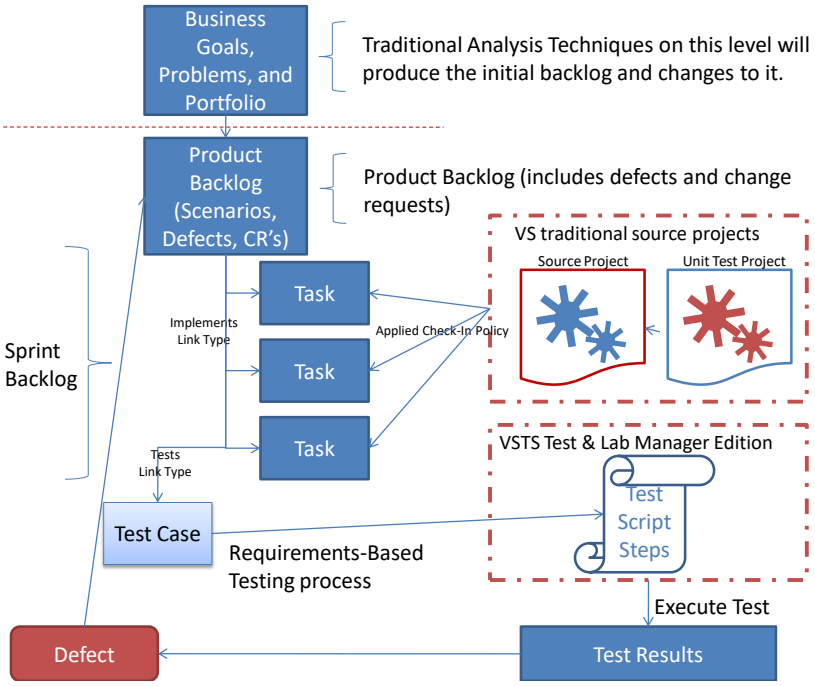


Figure 2: Scrum Traceability

Refer to the Requirements Traceability topic for specific details represented in this diagram.

Roles and Responsibilities

The table below depicts the roles that are required for a Scrum project.

Requirement or Evolutionary Level	Responsible Role
Functional Requirement (Scenarios or Stories)	Product Owner
Quality of Service	Scrum Team Member – a cross functional role that can be embodied in the developer, tester, Scrum Master, or any other member of the team. Agile projects aim at a collective code ownership, and, hence, *'All Assets'* ownership by each member of the team.
Test Case	Scrum Team member will develop test cases for functional requirements, scalability, load, and reliability requirements, Source Units and Components, and others depending on test types planned.
Source Code	Scrum Team Member

Requirements Attributes

The following attributes are just a few of the possibilities. They are listed here because they help in validation, progress toward completeness, and planning.

Conditions of Satisfaction – Description of what "Done" means for the specific task and its parent scenario.

Done / Not Done – Boolean value to represent completion.

Initial Estimate – Integer value usually represented in hour units.

Size (for product backlog) – Use T-Shirt sizing (small, medium, large, X-Large)

Change Management

Describe a process that holds all changes to requirements outside of the execution of the iteration. Product owners will negotiate a re-prioritization of product backlog items and take into account the changes that come during the iteration. This should not be detrimental to a project, due to the short nature of agile iterations. Rarely are requirements rendered completely obsolete by a change to one of them.

Workflow and Activities

The following is a list of the states relevant to agile work items.

1) Not Done (estimate and time remaining are the same)
2) Assignment → In Progress
    a. Time Remaining changes until task is done
3) Finish Task → Done (Work remaining = 0, and task artifacts test successfully)

Traditional Development Elements

Traditional requirements management planning is described above in the context of generic requirements management planning. With that said, however, a project following a traditional methodology will want to make sure to build a checklist for ensuring that requirements and their accompanying documents are all covered and planned for the execution of the project.

Requirements Traceability

Traceability is probably the most important dimension of requirements engineering in that it provides accountability to an application development team. In addition, it helps:

- Identify the source and importance of requirements
- Manage the scope of the project
- Manage changes to requirements
- Assess the project impact of a changes to a requirement or other element of the project
- Assess the impact of a failure of a test on requirements (i.e. test failure may mean the requirement is not satisfied)
- Verify that all requirements of the system are fulfilled by the implementation
- Verify that the application does only what it was intended to do
- Gives the developer a requirements context for his task

When implemented effectively, traceability will provide a map of the business stakeholders' needs to solution features, to product functionality, to its design against the application architecture, its tests, and, ultimately, the source code and data written to implement the solution. With such a broad reach through the artifacts of a project, it makes sense that traceability is a realization of an organization's defined development methodology.

When missing, or implemented at too macro a level, key change management functions become difficult and tedious. For example, when a business requirement is approved for a change to the scope of an application development project, without traceability, it becomes very difficult to estimate the scope of the change. Questions like "How many features and functions are affected by the change?", "Who needs to be notified of the change?", "What tests need to be updated as a result of the change?" etc… cannot be readily answered. In order to answer these and other change impact questions, traceability reduces the cost of the research and analysis because the information is already organized into an immediate hierarchy.

This topic within the Visual Studio ALM Rangers Requirements Management Guidance project will provide guidance to Visual Studio/Team Foundation Server users that desire an effective means for tracing high level requirements to their lower levels and to design, test, and source code elements. After reading the introductory paragraph above, many might think that this topic is heavy and laden with a lot of additional project documentation that an agile project will avoid. This is not the case. An agile team will see an implementation of traceability that supports the separation of product vs. iteration backlog management activities that are simple, yet powerful in achieving accountability and easing the burden of impact analysis.

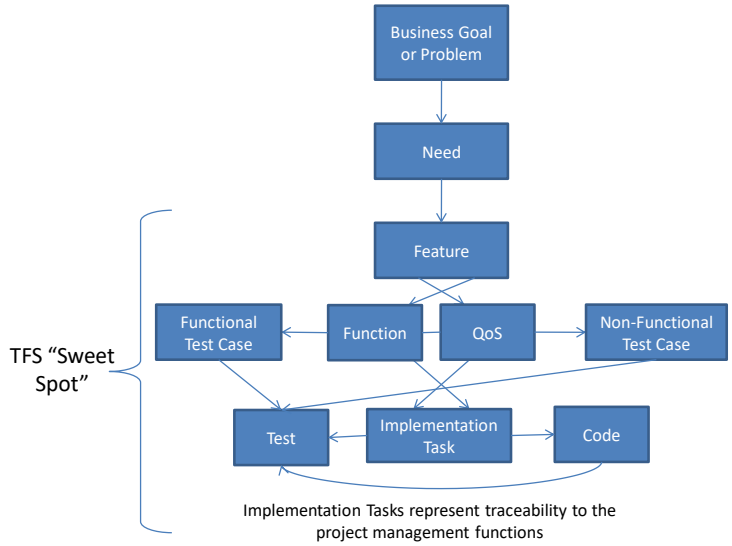
> **Note:** The guidance for requirements traceability found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Generic Requirements Traceability Guidance

Traceability Strategy

Below is the diagram of generic traceability that we presented in the topic on "Requirements Management Planning". It is shown here again to give a starting point for types of information that need to be tracked on any application development project. As stated in the planning topic, the "sweet spot" of the trace diagram below that is covered by Visual Studio and Team Foundation Server is highlighted by the elements linked just below the "Need" box.

Traceability Strategy



When implemented, each of these elements has a home in Team Foundation Server:

Artifact	Team Foundation Server Representation
Feature	Feature Work Item Type ***(MSF for CMMI)***
Function	Requirement Work Item Type with type = Functional (MSF for CMMI) User Story (MSF for Agile)
QoS (Quality of Service)	Requirement Work Item Type with type= "Quality of Service" (MSF for CMMI) User Story (MSF for Agile)
Test Case	Test Case Work Item Type ***(MSF for Agile and MSF for CMMI).*** This work item represents a container in Microsoft Test and Lab Manger. It includes Shared Steps that can be copied from test case to test case in order to represent frequently common functionality; i.e. logon procedures.
Test	Test Project (Solution Explorer) Test Projects in Visual Studio 2010 are specifically unit or performance tests at the developer's level or for

	"Test Driven Development" (TDD). Functional tests (manual and automated) are now an element of the Test Case work item whose test steps implement the procedure that can also be automated.
Implementation Task	Task Work Item Type (MSF for Agile and MSF for CMMI)
Code	Source Solution (Solution Explorer, Version Control)

Tests and Traceability

Visual Studio supports several types of tests. Each has its function and a different type of traceability. We can divide the tests into two major groups (although some tests are somewhat blurry and can be placed in either group)

- Low level tests – Tests created by developers and that are stored as project files (normally together within the same solution as the code). For example unit tests (code or database), automated tests written by the developer as a software verification and validation method. In this category we can also place load tests that are non-functional tests intended to stress an application to its capacity in order to evaluate its scalability and the ability to cope with a high load. The traceability of these tests is somewhat limited since their results are somewhat less interesting at a business level. Performance tests ride on the cusp of low level and system level testing. They are low-level in that they are built using test projects in Visual Studio. They are system level in that they verify the non-functional implementation of a non-functional or quality of service requirement type.
- Functional testing – Functional tests that are directly related to the business level requirements. These can be system tests that test if a user story has been implemented as specified or a UAT test that checks if a given business feature has been implemented as requested at the business level.

  Low level tests have a low level of traceability since they are more important at a development level. This doesn't mean they are not traceable, it simple means they are not done at a business level. For example unit tests can be linked to work items and test results (either unit tests or load testing tests) can be linked to work items (eg: link the failed execution of a test to a work item of task bug for easier resolution). In the current version the traceability of low level tests is stronger at their execution instead of their definition. This allows you to check the trend in test execution (e.g.: number of passed/failed tests over time and the regression of tests) and trace at the code level the percentage of code that is tested (code coverage).

  The level of traceability of functional tests is much higher since this is the kind of linkage that is very interesting at a business level. We cannot only link tests to tasks but more importantly we can link tests to user stories. This allows us to get full traceability and clear understanding at the business level which features are tested (and their percentage) and which features do we still need to concentrate our tests so we have full coverage of testing at the requirement level.

When defining test cases you cannot only define the test itself but also the steps (which can be shared with other test cases) that compose a given test. This way you trace which tests have passed and failed as well as which steps have worked and not worked during a given test run.

Typed Links between artifacts

In Team Foundation Server different links can be created between those artifacts:

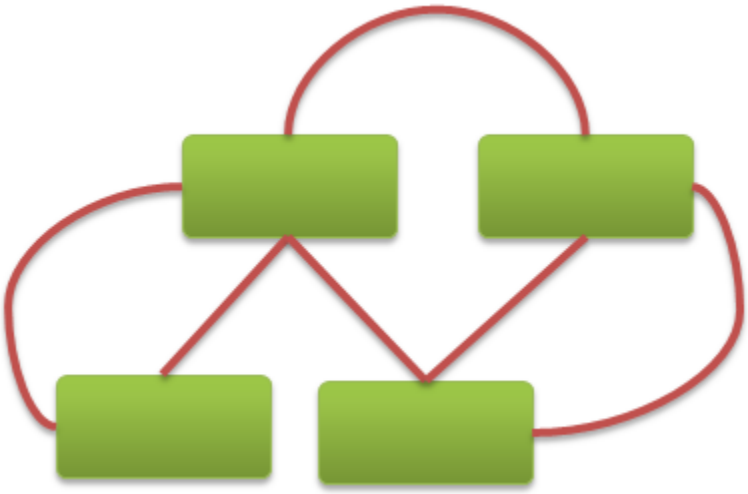
- Changeset – A link between work item and changeset
- Versioned Item – A link between work item and a folder or path in source control
- Work Item – A typed link between two work items (see below)
- Hyperlink – A link from a work item to a URL
- Test Results – A link from work item to results from test execution

Note that in Visual Studio 2010 links to work items are now capable of providing semantics for directionality and relational information to denote parent, child and sibling relations for linked work items. This capability provides greater traceability as it becomes trivial to see how one work item hierarchically relates to another. This provides information to understand how a requirement may have been refined into multiple "child" requirements from an initial requirement that was too broad.

Work item links have types and follow a topology. The available topologies are:

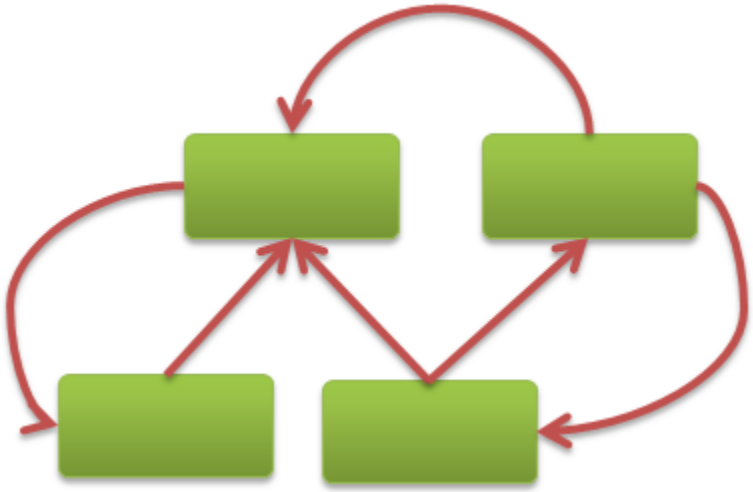
- Network

  Link types of this topology have essentially no rules and no directionality. You can have circular relationships, and the link looks the same from both sides.



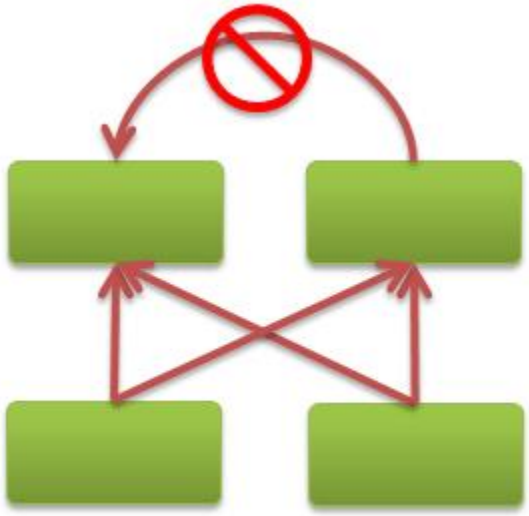
- Directed Network

Link types of this topology are like Network links, except there is directionality. You can specify a name that appears at each end of the link. In other words, the link looks differently depending from which side you view it.



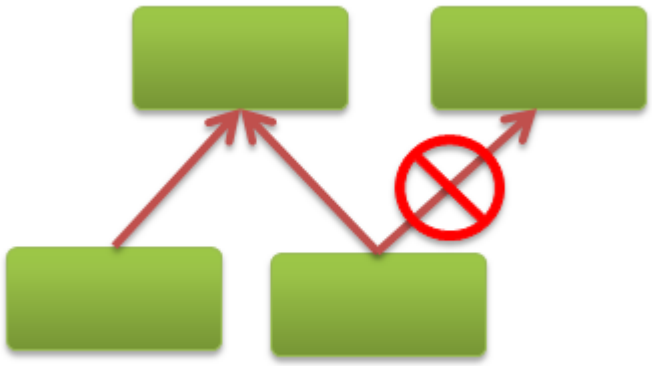
- Dependency

  Link types of this topology are like Directed Network links in that they have directionality, but an additional constraint to prevent circular relationships.



- Tree

  Link types of this topology are essentially trees, it enforces a one-to-many relationship and doesn't allow circularity.



Link types that are defined by the system:

Forward Name	Reverse Name	Link type reference name	Topology
Successor	Predecessor	System.LinkTypes.Dependency	Dependency
Child	Parent	System.LinkTypes.Hierarchy	Tree
Related	Related	System.LinkTypes.Related	Network

Link Types Defined by MSF Process Templates:

Forward Name	Reverse Name	Link type reference name	Topology
Tested By	Tests	Microsoft.VSTS.Common.TestedBy	Dependency
Test Case	Shared Steps	Microsoft.VSTS.TestCase.SharedStepReferencedBy	Dependency

These links are used to reflect implementation relationships which enable a work-break-down structure for features, user stories and tasks:

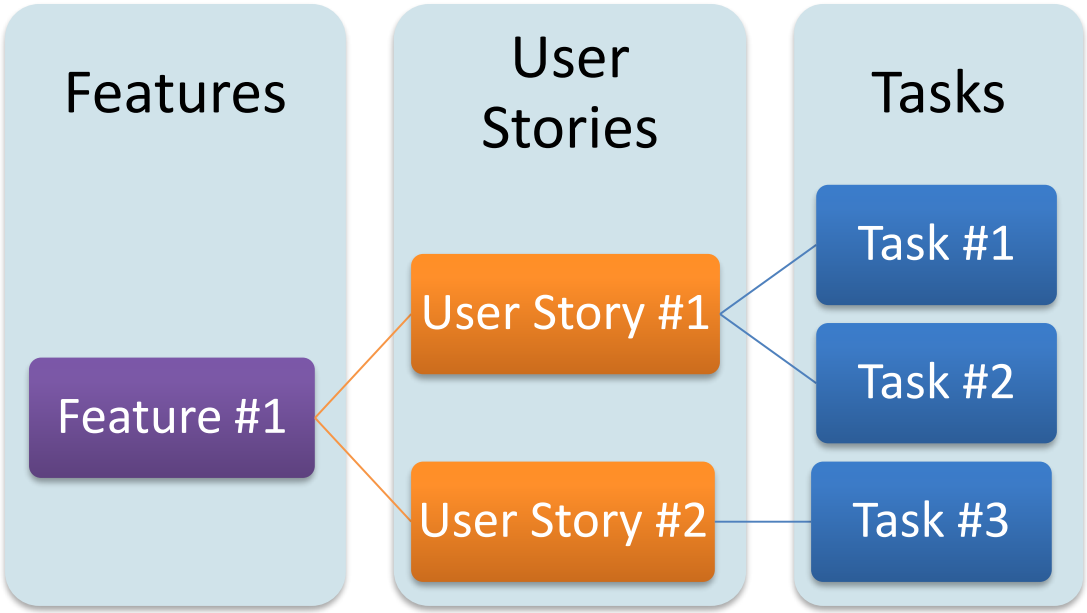


Figure 3 – Feature → User Stories → Tasks

Test definitions can be linked to features or user stories based on the testing level (e.g. user acceptance test definitions are linked to features whereas system test cases are linked to user stories):

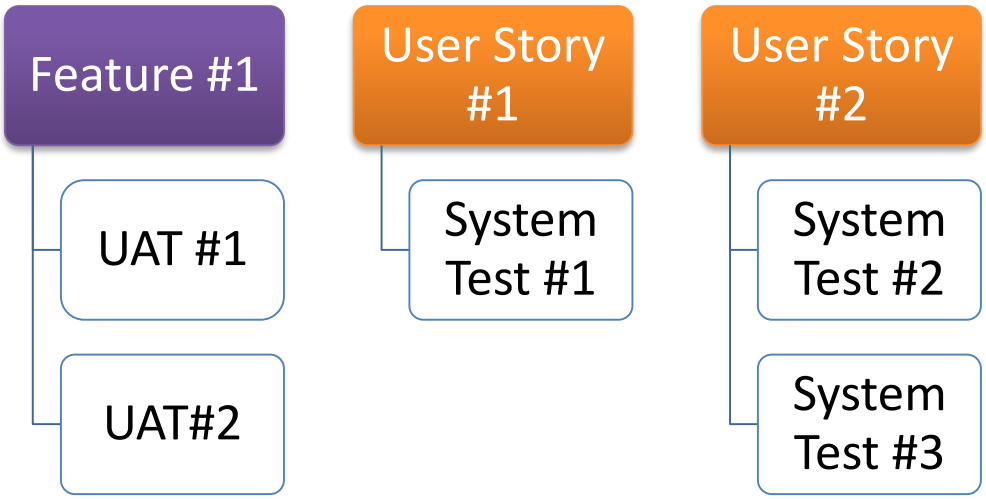
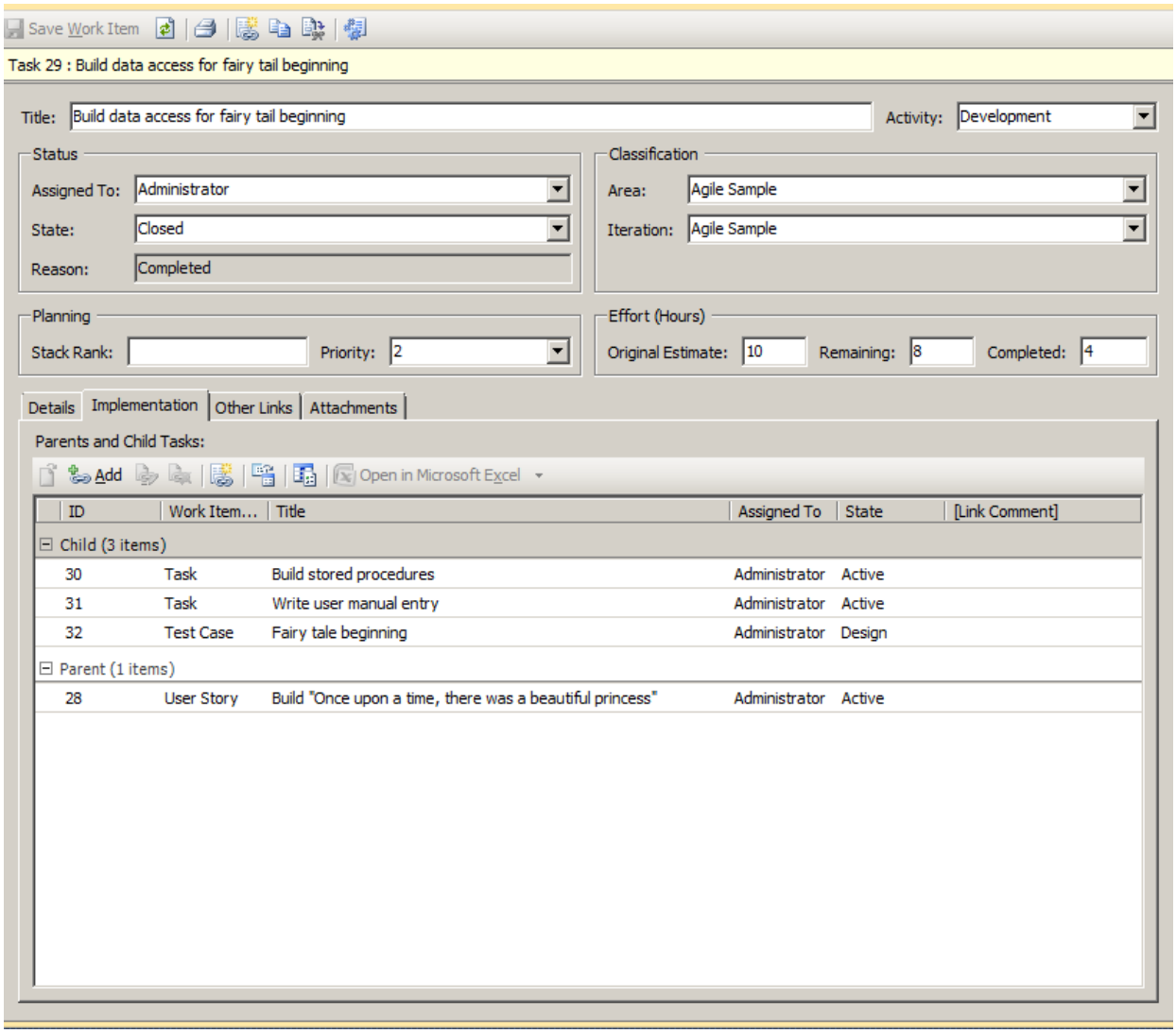
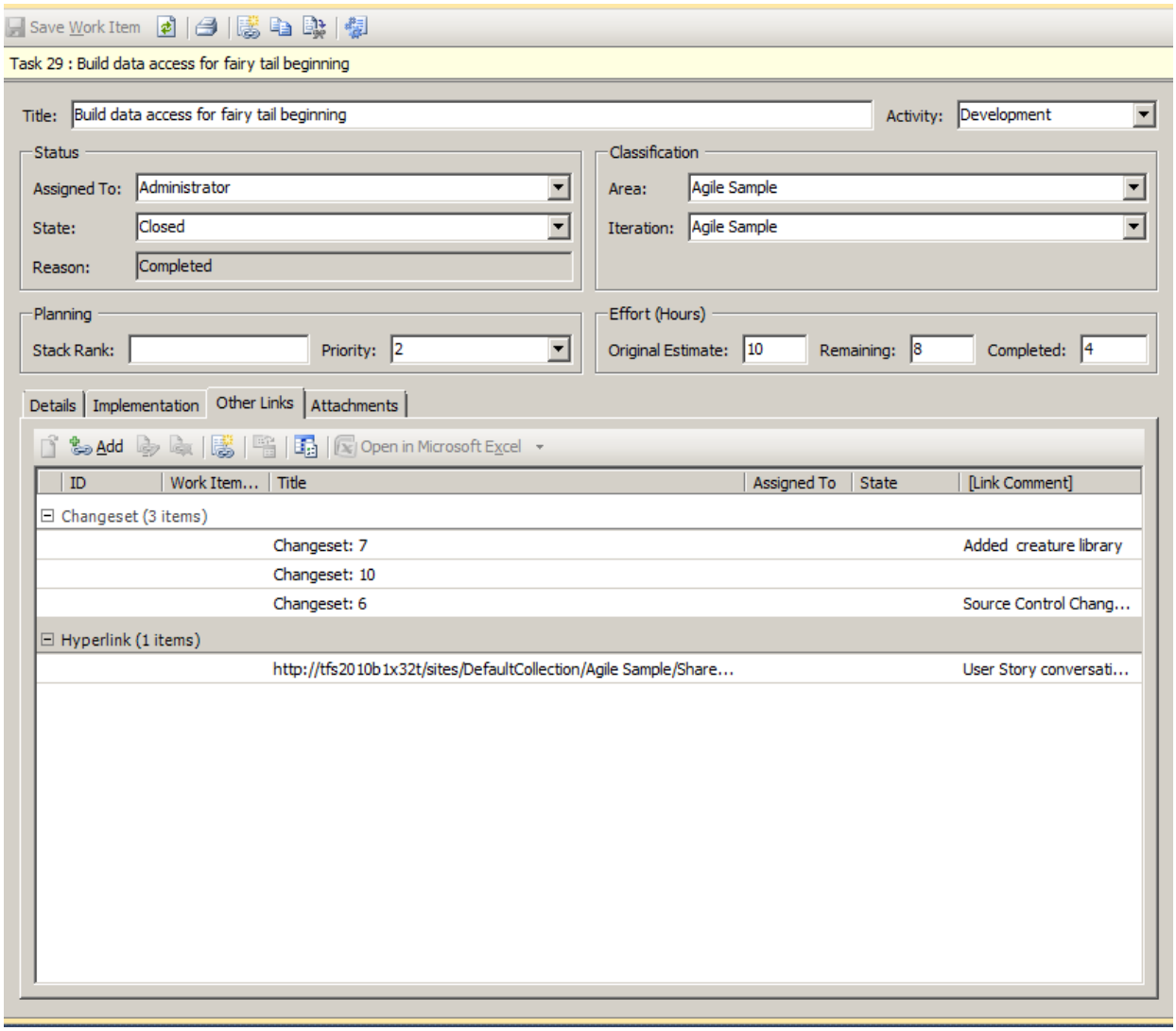


Figure 4 - Feature → UAT; User Story → System Test

Links are displayed in a configurable work item control.

Links can be filtered and are grouped by type.


Save Work Item
Task 29 : Build data access for fairy tail beginning
Title: Build data access for fairy tail beginning
Activity: Development
Status
Assigned To: Administrator
State: Closed
Reason: Completed
Classification
Area: Agile Sample
Iteration: Agile Sample
Planning
Stack Rank:
Priority: 2
Effort (Hours)
Original Estimate: 10
Remaining: 8
Completed: 4
Details
Implementation
Other Links
Attachments
Parents and Child Tasks:
Add
Open in Microsoft Excel
ID
Work Item...
Title
Assigned To
State
[Link Comment]
Child (3 items)
30 Task Build stored procedures Administrator Active
31 Task Write user manual entry Administrator Active
32 Test Case Fairy tale beginning Administrator Design
Parent (1 items)
28 User Story Build "Once upon a time, there was a beautiful princess" Administrator Active



Using links for documentation

Documentation can be generated using the linking information from work items. In the trace diagram shown earlier, you can see an arrow drawn from each requirement type or project artifact to a section in the generic requirements specification. So, rather than document each section of a required document manually and then manage versions of the document and juggle the various "hands" that need to touch it simultaneously, Team Foundation Server users can produce the document as a report of the state of the requirements at any time during the project by developing a report using SQL Reporting Services, also a core component of Team Foundation Server. This notion will save a lot of time and improve the quality of the deliverable because it can be formatted once and be used by all projects on-going. It also allows the set of requirements to be developed collaboratively by several people who can independently write or update requirements without the need to wait on each other. And changes to the requirements in any phase of the project will update the documentation automatically so it does not get outdated.

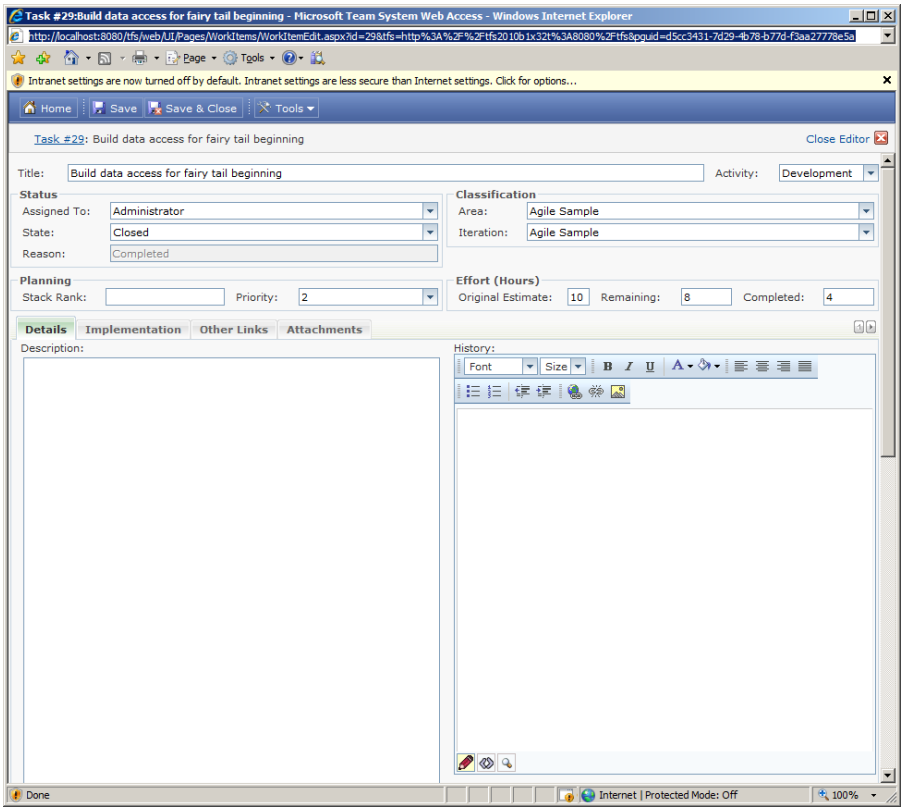
If documentation is too detailed to be specified in a work item, the work item can be a high level descriptor and placeholder for the more detailed information. A document, PowerPoint slide(s), Visio diagrams, Excel spreadsheets, and other file-based requirements details can be stored in the SharePoint portal of a team project. Then, using the link of a work item, these files can be linked to a work item using the "URL-type" link. Within the document itself the ID of the work item as well as a URL to the Web Access (TSWA) for the work item can be listed to provide bi-directional traceability.

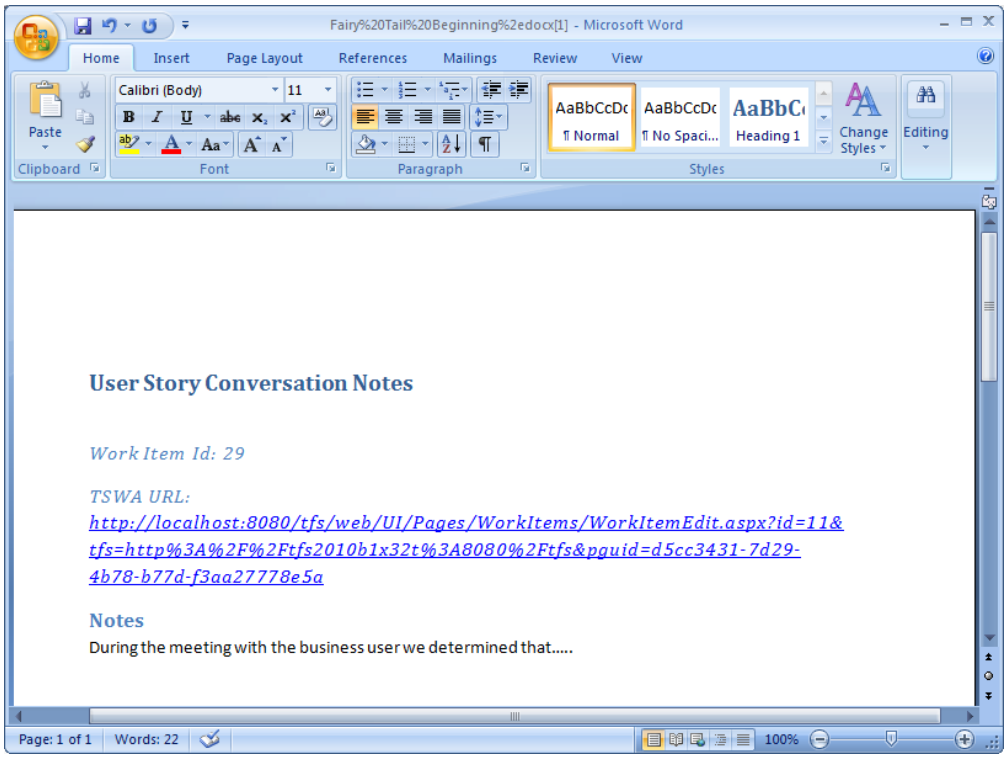
From an execution standpoint, any team member that needs to work on or understand the details of a requirement need only have the requirement assigned to them. They open the work item in a Visual Studio Work Item Query (i.e. All Work Items for which I am assigned) and then navigate to the link that specifies the details. Likewise if a user is browsing a document stored within the project portal the link to the TSWA page for the work item will allow them to view the state of the work item and related artifacts.

Customizable Traceability

The diagram and description given above demonstrate capabilities that exist without any customization to Team Foundation Server. Because of its flexible nature, users can either customize one of the two process templates (MSF for Agile and MSF for CMMI) that come with the native install of Team Foundation Server, download and customize a 3rd party process template, or create their own.

As such, a process template can be configured to have work items that represent any methodology's requirement types and nomenclature as well as the relationships between work items using the robust linkage capabilities available with Visual Studio 2010.


Task #29:Build data access for fairy tail beginning - Microsoft Team System Web Access - Windows Internet Explorer
Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...
Home
Save
Save & Close
Tools
Task #29: Build data access for fairy tail beginning
Close Editor
Title:
Build data access for fairy tail beginning
Activity:
Development
Status
Assigned To:
Administrator
State:
Closed
Reason:
Completed
Classification
Area:
Agile Sample
Iteration:
Agile Sample
Planning
Stack Rank:
Priority:
2
Effort (Hours)
Original Estimate:
10
Remaining:
8
Completed:
4
Details
Implementation
Other Links
Attachments
Description:
History:
Font
Size
Done
Internet | Protected Mode: Off
100%


Fairy%20Tail%20Beginning%2edocx[1] - Microsoft Word
Home
Insert
Page Layout
References
Mailings
Review
View
Calibri (Body)
11
Paste
Clipboard
Font
Paragraph
Styles
Change Styles
Editing
¶ Normal
¶ No Spaci...
Heading 1
User Story Conversation Notes
Work Item Id: 29
TSWA URL:
http://localhost:8080/tfs/web/UI/Pages/WorkItems/WorkItemEdit.aspx?id=11&tfs=http%3A%2F%2Ftfs2010b1x32t%3A8080%2Ftfs&pguid=d5cc3431-7d29-4b78-b77d-f3aa27778e5a
Notes
During the meeting with the business user we determined that.....
Page: 1 of 1
Words: 22
100%

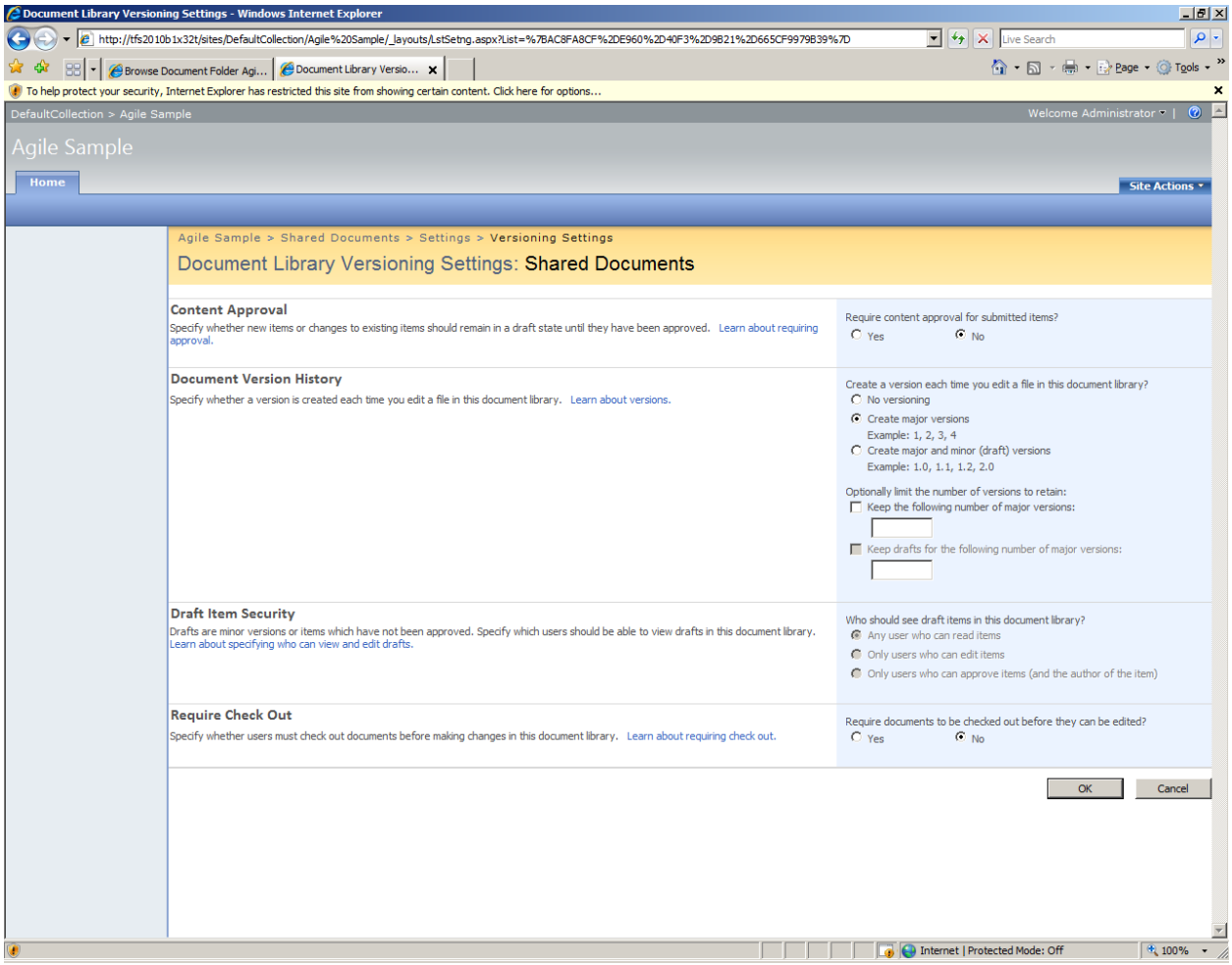
Governance

Compliance to industry governing organizations or legal entities is too large of a topic to cover in this guidance, but, using Team Foundation Server for traceability can help a development team achieve the following:

- CMMI Level 3 capability for Requirements Management (ReqM) and Requirements Development (RD)
- Food and Drug Administration (FDA) compliance to regulation CFR-21, part 11 that specifies digital signatures
- ISO 9000 compliance for auditable compliance to contracted development
- Sarbanes-Oxley compliance for audit ability of financial transactions

There are other models, but the main idea to understand is that Team Foundation Server can provide traceability that achieves accountability. The models for development maturity or government regulations were established to protect innocent people from shoddy business practices. Traceability helps protect these people and Team Foundation Server is the technology that achieves it.

Enable the versioning features within SharePoint to track the history of changes to the documents. In conjunction with the full audit trail of work items maintained by Team Foundation Server this will provide additional audit capabilities to all artifacts of a project.



The "Infamous" Traceability Matrix

Before moving on to traceability guidance specific to agile or traditional projects, it is important to address the one trace artifact for which compliance auditors look to provide evidentiary support for a project's traceability. The idea of a traceability matrix is important in that it provides a "beginning to end" mapping of the work performed on a project.

For CMMI and ISO compliance, that means demonstrating that the project can trace all business requirements to functional requirements to technical requirements to tests and, ultimately, to source code and defects. The following table is an over-simplistic demonstration of a typical manual or "Spreadsheet" implementation of a trace matrix:

BID	Business Req.	FID	Functional Req.	TRID	Technical Req	TCID	Test Case	Source File
\\sharedlibrary\project_doc\Requirements\My Project Business Requirements.doc								
BR-1	The application shall improve the customer experience as they buy our products, giving us higher cust sat and increase our revenues.	UC-1	\\sharedlibrary\project_doc\Requirements\Customer_Self_Service_Purchase_Use_Case.doc					

				TR-1	Support this function for 200,000 concurrent users without performance degradation	TC-1	Verify transaction duration for single user and compare against the average duration for 200,000 simultaneous transactions	\\my_workspace\source\my_source.cs \\my_workspace\source\my_source2.cs Etc…
				TR-2	Develop operational guidance and troubleshooting for tech support.	TC-2	Verify blah, blah, blah	\\my_workspace\source\my_source.cs \\my_workspace\source\my_source2.cs Etc…

This manual implementation of a trace matrix presents significant difficulties to collaborating teams:

- As a spreadsheet, only one person can maintain it at a time. There are government organizations that hire full time resources whose only job on a project is to maintain this traceability. There job includes going to each developer, business analyst, project manager, etc… and gathering their changes to include in the matrix and provide impact analysis to the team. This expense seems (in this author's opinion) to be a waste of government funds (a.k.a. Our Tax Dollars).
- The spreadsheet is almost useless for tracking down the impact of a change or areas affected by a defect. In a 6 month project performed by a team of 12-15 resources (so, relatively small) the trace hierarchy can include over 20,000 trace links. This spreadsheet becomes difficult to navigate in support of impact analysis. The reality on project teams using this matrix is that the developers never even read it.
- Because it is developed manually, usually as an afterthought, the accuracy of this report is very low. On some government projects, this report is put together with high level data that is not comprehensive at the end of each phase of a gated project (waterfall). Because the size of the document would require days of analysis for a check of the accuracy, auditors list the document as compliant without performing the analysis.

As can be seen in the screenshots of the task work item above the full traceability to each of the artifacts as identified in the sample spreadsheet are easily identified from within the work item itself and is maintained within the Team Foundation Server.

Team Foundation Server Reports using Work Item Queries in addition to the OLAP Cube

The following list represents reports that should all be generated using data that exists in Team Foundation Server's OLAP Cube (not native SQL) or in the Work Item Store:

- **Business Requirement to System Requirement trace** – demonstrate that you have comprehensive analysis of business requirements and at least have one functional requirement (MSF for CMMI) or User Story (MSF for Agile) for each BR. This report will (more

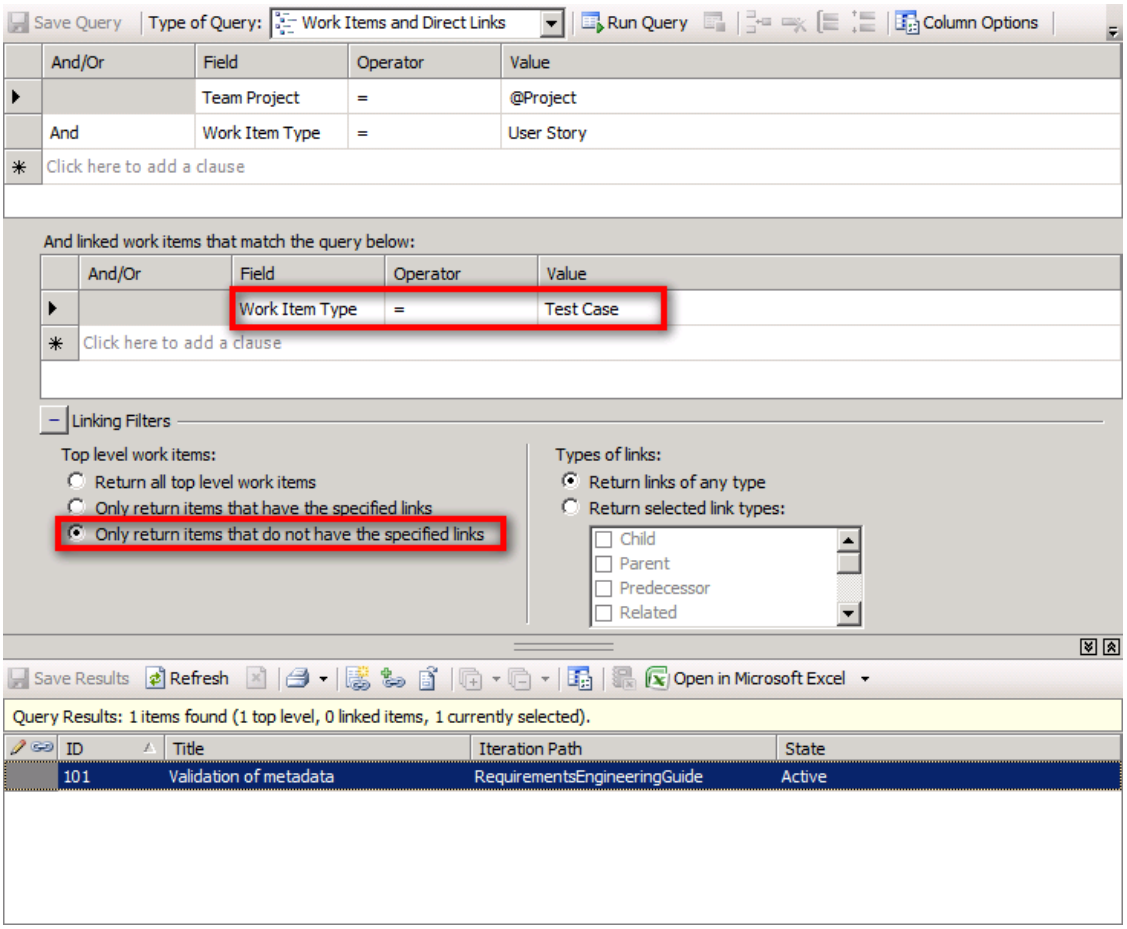
importantly) demonstrate where you are missing a trace. It points out where you have more analysis work to do

- **Functional Requirements to Technical Requirements** – this one demonstrates a similar coverage at the next level down. The debate here is "what is a technical requirement?". Following industry standards established by IEEE, CMMI, and ISO, we are going to use the definition of a technical requirement as a non-functional requirement that identifies qualities of service (Performance, Reliability, etc.) Usability (User Documentation, User Experience, Help, Training, etc…), Constraints (Enterprise Architecture decisions like OS, DB system, middleware, Application Architecture platform, etc…), Operations and Support (Volumetrics, Offline Processing, Failover, Business Continuity, etc…) Some communities think that this covers design documentation, but that opinion is an industry-wide misconception.

- **Functional and Non-Functional requirements to Tasks** – demonstrate that you have activity planned for each of your requirements. In the sample attached here, I actually use this data to reflect the completion status of the tasks. Conchango's template has built bits into the event handler that will roll up changes to the work remaining on a task (sprint backlog item) to the scenario (product backlog item) automatically (cool stuff).

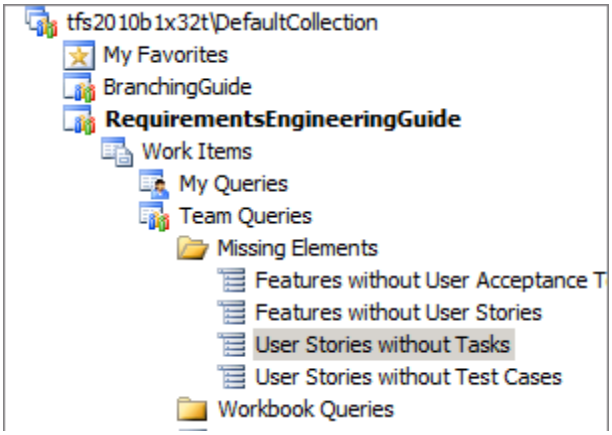
- **Tasks to Change Sets, Tests, and Defects** (as well as the reverse) – this one is where the 'rubber meets the road' in terms of why FDA compliance was defined. It gives you accountability where source code quality is concerned. When you have a change set that is not linked to a task (or a requirement), it indicates where someone may have made a change to source code that could be related to the "safety critical" nature of medical devices. In financial systems, it is where you identify malicious attempts to steal money (I like to refer to this as the Superman 3 Movie scenario or Office Space).

- **Missing elements** – Traceability should not only visualize the relation between the different artifacts, but it should also make sure that the items have their desired trace elements. For example, when a feature doesn't have a User Acceptance Test case identified, this query can be filtered to show those features. This doesn't ensure comprehensive identification of all positive and negative tests, but it does show where no work to identify tests was performed. Based on the links it is easy to set up a query for each of the items showing those where corresponding sub items are missing.

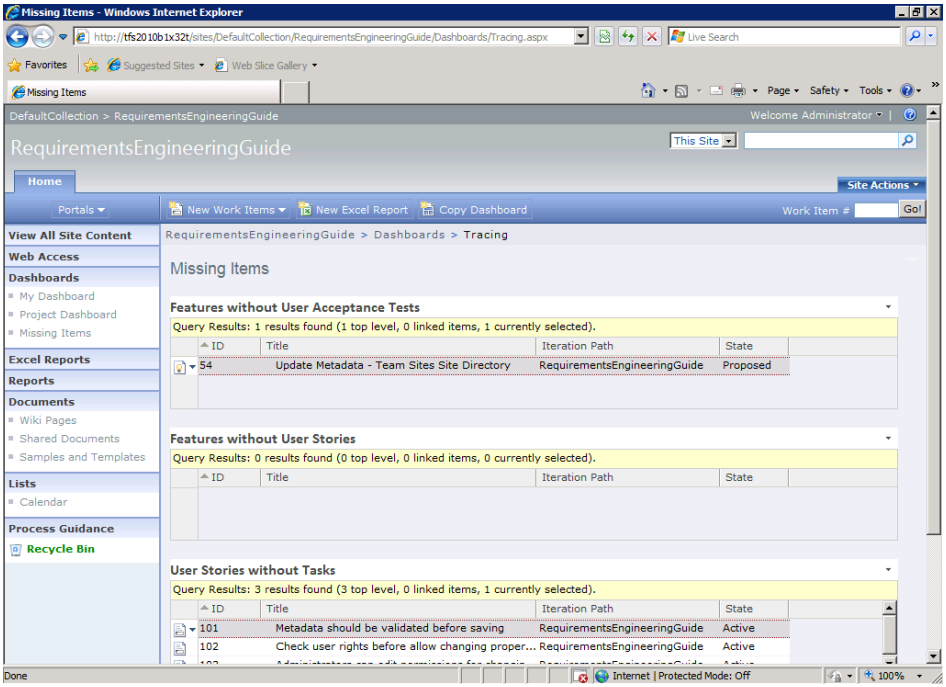


Examples for helpful Queries are:

1. Features without User Acceptance Tests
2. Features without User Stories
3. User Stories without Tasks
4. User Stories without Test Cases



The Query runs against live data so it returns the current state when refreshing. This is helpful if you fill the gaps and define the missing items. Then you can rerun the Query to see what is still left to do. The Query can be run inside of Visual Studio, Excel or the SharePoint Dashboard.



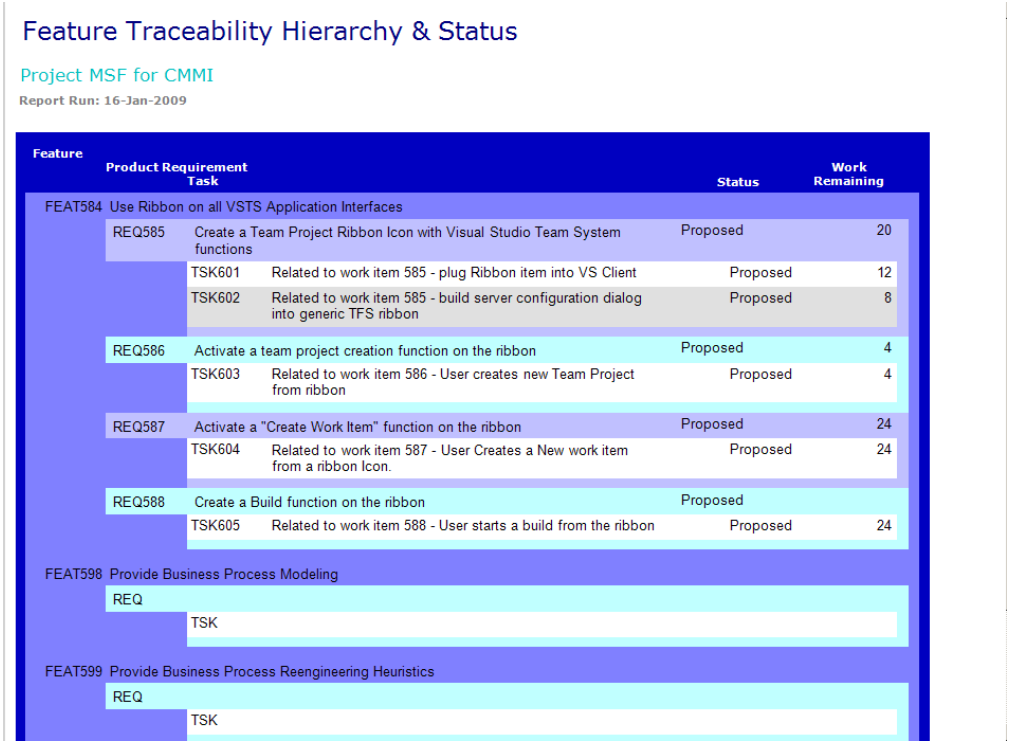
NOTE: Team Foundation Server Reports Using Native SQL

Sometimes, a trace report is valuable only if it can demonstrate multiple levels of a hierarchy. For example, the ability to report on a set of Business Features, showing the scenarios that will realize their implementation and the tasks identified to implement the solution, complete with their status of 'done-ness' and work remaining to complete them. This report is not readily generated using the Team Foundation Server OLAP Cube. Though it is possible, the same report can be generated with more flexibility by using Native SQL against the data warehouse data. The following table represents such a query. The column on the left explains the sections of the query and where it should be edited to reflect any project's specific trace hierarchy and attributes.

- **Feature to Requirement to Task Trace → Feature Progress Summary Report** The following report sample demonstrates the pieces of the query to trace 3 levels within a tree of traced work items. The value of this report is that it demonstrates in a single report an organization through multiple milestones of a project. Particularly for a traditional development lifecycle (waterfall), it can demonstrate the traceability from the business to the functional to the technical requirements. It's at that milestone that final development estimates are usually made. For an agile implementation, the 3-level traceability report

provides a view from the product backlog to the sprint backlog to the tests. It is a single view of the hierarchy of elements that can give progress toward completing a plan.

The report sample generated by the above query looks as follows:



Feature Traceability Hierarchy & Status

Project MSF for CMMI

Report Run: 16-Jan-2009

Feature	Product Requirement Task		Status	Work Remaining
FEAT584	Use Ribbon on all VSTS Application Interfaces			
	REQ585	Create a Team Project Ribbon Icon with Visual Studio Team System functions	Proposed	20
	TSK601	Related to work item 585 - plug Ribbon item into VS Client	Proposed	12
	TSK602	Related to work item 585 - build server configuration dialog into generic TFS ribbon	Proposed	8
	REQ586	Activate a team project creation function on the ribbon	Proposed	4
	TSK603	Related to work item 586 - User creates new Team Project from ribbon	Proposed	4
	REQ587	Activate a "Create Work Item" function on the ribbon	Proposed	24
	TSK604	Related to work item 587 - User Creates a New work item from a ribbon Icon.	Proposed	24
	REQ588	Create a Build function on the ribbon	Proposed	
	TSK605	Related to work item 588 - User starts a build from the ribbon	Proposed	24
FEAT598	Provide Business Process Modeling			
	REQ			
	TSK			
FEAT599	Provide Business Process Reengineering Heuristics			
	REQ			
	TSK			

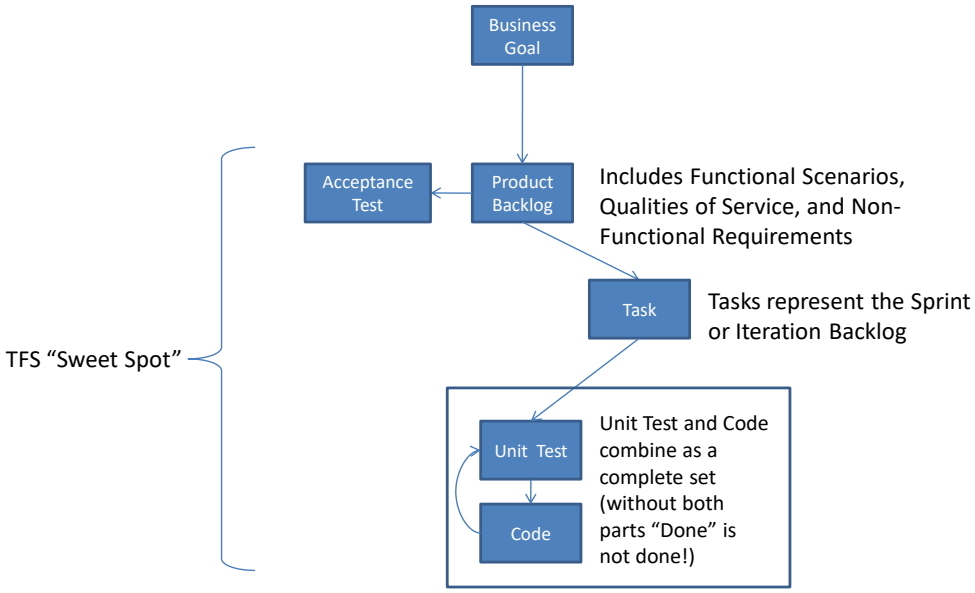
Notice in this report that Features are at the highest level in the hierarchy. Within each Feature, the report displays lists of Requirements to which they are directly linked. Within each Requirement, the report displays lists of tasks to which the requirements are directly linked. In this report, we've demonstrated the work remaining attributes that roll up from the tasks to the requirements. The sum of work remaining on the requirement is manually calculated in the MSF for Agile and MSF for CMMI templates. In the Conchango process template, this data automatically rolls up to the requirement level.

Another thing to point out is that when a Feature is not represented by any requirements, a hole in coverage is readily apparent. Planning that has not been accomplished for the implementation of each requirement is also readily apparent.

Traceability Guidance for Agile Projects

Agile projects will take a streamlined approach to traceability. Using the slogan from the Agile Manifesto, "Working Software over Comprehensive Documentation", an agile team is looking for a trace hierarchy that is minimalist in approach. With that said, however, there needs to be enough traceability to documentation that ensures that the team can effectively cover all of their requirements and determine the impacts of changes or new requirements on the existing project. The following diagram demonstrates a minimalist approach that maps to the generic trace hierarchy above.

Agile Traceability



Traceability Guidance for Traditional Model Projects

Traceability for traditional development projects is no different than for agile projects. The requirements types covered in the generic section of this document account for the traditional model. If your methodology requires a different trace hierarchy, ensure that the appropriate requirements types and trace detail are planned. Refer to the Requirements Management Planning topic area for more guidance.

Analysis and Breakdown

"Analyses are performed to determine what impact the intended operational environment will have on the ability to satisfy the stakeholders' needs, expectations, constraints, and interfaces. Considerations, such as feasibility, mission needs, cost constraints, potential market size, and acquisition strategy, must all be taken into account, depending on the product context. This, all in addition to a definition of required functionality, includes all specified usage modes for the product." – CMMI, Guidelines for Process Integration and Product Improvement, Chrissis, Konrad, Shrum.

Analysis and Validation is performed to:

- Establish Operational Concepts and Scenarios (Product Requirements – e.g. user goals and context diagrams, Product Component Requirements – e.g. technical constraints and operational concepts)
- Establish and maintain a definition of required functionality (Functional Architecture, Activity diagrams and use cases, object-oriented analysis with services identified)
- Analyze Requirements (Requirements defects/volatility, Requirements Changes, Technical Performance measures, Assessment of risks)
- Validate Requirements with Comprehensive Methods (Record of analysis methods and results)

The Analysis and Breakdown of requirements relies explicitly on other topic areas in this guidance. The topic area begins in the Requirements Engineering lifecycle at the point the initial business requirements are gathered. Analysis of the business requirements coupled with the appropriate elicitation techniques, described in the Elicitation Guidance topic area, will allow the team to derive functional requirements and initial project estimates. Further analysis and elicitation of the functional requirements in the context of the business requirements will allow the team to derive quality of service requirements and refined project estimates based on task analysis.

With validation techniques using checklists at each level of the evolution, the quality of the requirements increases. This quality is further gained because the team is in a better position to define tests for acceptance at each level of the requirements evolution.

> **Note:** The guidance for requirements analysis found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Analysis and Breakdown Process

For each level of the evolution, the team goes through a series of steps that result in requirements work items and detail that should be captured in Team Foundation Server.

Business Level Analysis

The goal of business level analysis is to identify the business level requirements for the engagement or software development effort. These requirements will be stored in Team Foundation Server as "Feature" work items. Any detail that cannot be readily stored on the work item will be stored in SharePoint and linked to the feature. This level of analysis is typically performed before project teams are assembled on waterfall projects, or before an initial product backlog is created on agile projects.

Analysis and elicitation techniques should flesh out the business level as feature work items and validation will be provided by User Acceptance type Test Cases linked to them.

Business Level Analysis begins with a business problem or goal. Because business modeling and business case development are topics in which development teams don't typically participate, our guidance assumes that a business case exists and preliminary funding for the project has been procured.

With that, the business case (or similar instrument) provides the details describing the problem to be solved. It should already contain a preliminary identification of a solution and estimates for delivery. If the customer doesn't provide this, they will at least have done the preliminary work on their own and the team still should be able to start with something that the customer should be willing to share, such as an open discussion of the problem; not necessarily how much money they have allocated. The analysis at this level should review the problem to ensure that all stakeholders have been identified. In addition, the root cause of the problem or business goal should be reviewed and then the features of a solution need to be identified. The bulk of the work at this level will be through brainstorming the potential correct features that solve the problem or achieve the goal. These features need to be specified as work items in Team Foundation Server with additional detail in documents, spreadsheets, or diagrams stored in WSS and linked to the work items via hyperlink links. The following diagram demonstrates the process:

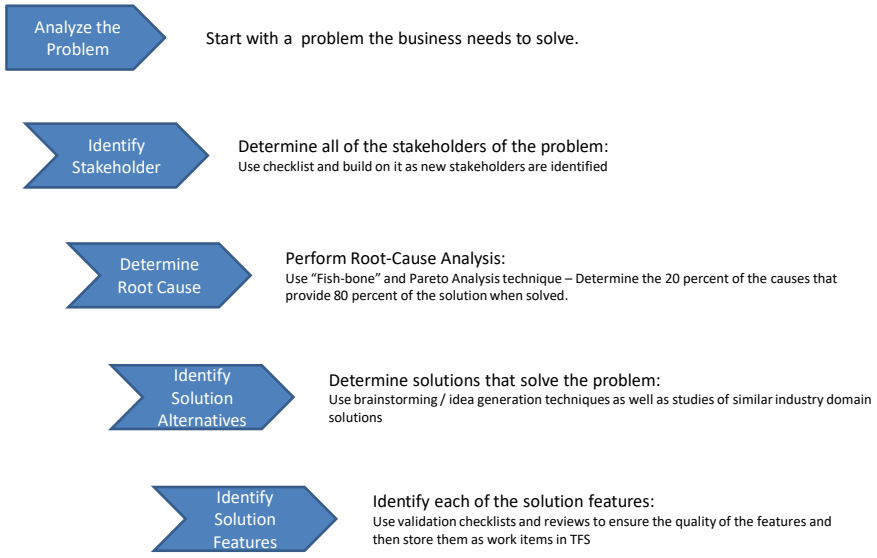


Figure 5: Business Level Analysis

This will most likely require the most creativity of elicitation and validation techniques of all the requirements analysis activities in this guidance. This is because the user's business requirements will come to the team in a variety of means. For the purposes of this document, we'll categorize 3 formats that should fit most needs of the team performing the analysis.

1) **Customer Formal Request for Proposal** – In this document, the customer will provide an organized (maybe not) list of categories that they need to be provided in a scope section of the document.
2) **Customer scope/vision/charter Document** - In this document, the customer will use a template from their own methodology. Though it will likely have been developed using a template from a common methodology, like RUP, the rigor applied to the scope and the style of capturing the features and needs could be in varying degrees of quality.
3) **An e-mail from the customer for a scope discussion** – In this situation, you are afforded the opportunity to drive the requirements format from your own perspective and can use planned elicitation techniques to identify the scoped features of their solution.

Whichever format you receive, you will need to engage with the customer representative(s) and validate their requirements and possibly gather them in the process.

At a high level, here is the process for Business Analysis:

1) **Analyze the Problem** – The problem for the customer may already be in one of the documents that they provided the team. Often times it is written in a Scope or Vision document as the "Problem Statement".

2.1 Problem Statement

Problem:	Team site owners may need to modify selected metadata stored on the site to more accurately reflect the site's purpose. Modifiable data includes the company name (for external sites), the site type, site title, site description, function, subfunction, business unit, region, country and whether or not the site has a 'legal hold code' associated with it.
Affects:	Company A|Team Site users Client Support
Impacts (if no solution is provided):	Inaccurate metadata could impact search, metrics used by SharePoint governance.
Successful Solution:	Accurate metadata. Increased ability to find sites. Meaningful metrics for management.
Results:	Users will be able to find their sites easily. Management will have more accurate data on sites by function, region, etc.

Figure 6: Sample Problem Statement from an MCS Sample's Scope/Vision Document.

The problem statement will often times provide detail that can be entered in the "Problem" field of the Quality Gates Tab of a new Feature work item.

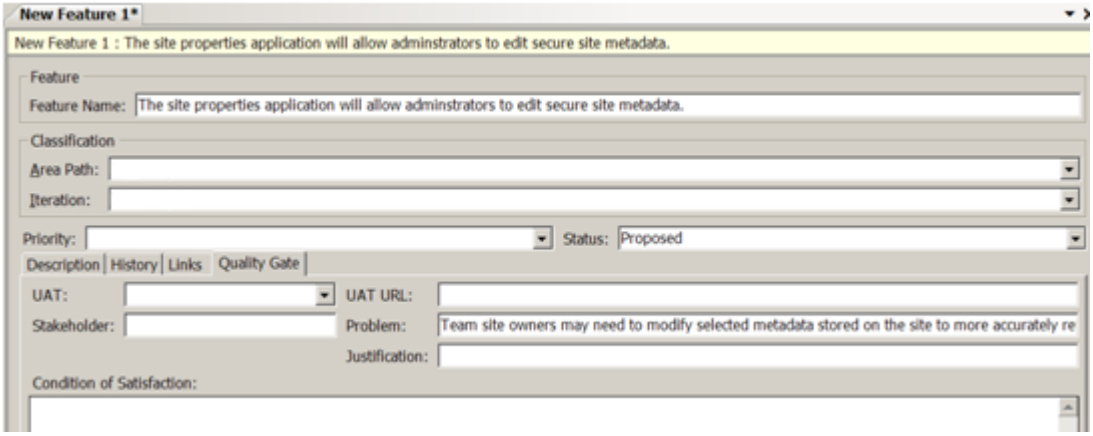


Figure 7: Feature sample showing problem.

2) **Identify all stakeholders**: - Determine from the customer the following people and their goals:
   a. Problem Owner: Who needs to have your solution implemented? This person or people will be the ones that give you the most business-level requirements for the solution.
   b. Financial Sponsor: Who is paying for your solution? This person or steering committee can care little about the actual solution and its requirements so much as they are a

champion for the problem owner and need to make sure that the problem owner is getting their needs fulfilled.

c. Subject Matter Experts: These are the people that know and understand most about the business and are going to be assisting your team in defining the feature requirements for the solution. They might also provide business guidance to your team as you work with the user stakeholders to identify system level functional requirements.
d. User Stakeholders: Who are the people that will use the solution? Often times they are also the Problem Owner, but sometimes they are not. User stakeholders, will provide the functional requirements and their validations during feature analysis (next section).
e. Support Stakeholders: These are people that will either own the operational support for your solution once it's in place, or they will provide infrastructural support to the team early on and throughout the development effort.
    i. Operational Support will provide requirements for getting the solution deployed in the production environment and any instrumentation requirements that your team will need to fulfill with your solution. (i.e. Run-Time Log File Writing or Run-Time Analytics, etc…)
    ii. Infrastructure Support will provide hardware, operating system software, utilities, tools, and vendor software (like database services and client installations) and licenses for your team. As such, they are going to define constraints for the environment for which you'll need to comply.
f. IT Stakeholders: If not the operational or infrastructure support people, these stakeholders will provide architecture, database, communications and interface, and security requirements and constraints to which your team will need to conform.

Storage of Stakeholders: We did not create a work item for this information, so the team will need to capture it in a document and store it in the SharePoint library related to the Team Project. In the templates folder of the Team Project generated by the MCS Process Template, navigate to the Documents Library → Templates folder and copy the Stakeholder Profile Template.doc.

Enter your stakeholders, using the above information (a-f) into the template and identify any other stakeholders that are provided by the customer. Their profile will define their role.

Project Stakeholders

<Below is a sample of some of the stakeholders for a project. Remove the samples after filling in with the project team's stakeholders.>

Stakeholder Name	Role / Profile Title	Description	Stake in Project
Chuck I. Orwell	CIO	The CIO owns the information technology of the organization. Enterprise architecture and technology decisions will ultimately be approved, funded, and authorized by this person.	Chuck wants to provide better automation to his business customers.
Paula Problema	Problem Owner	The problem owner need the development team to provide a solution that removes their problem	Paula needs your solution to ...
Scotty Expy	Subject Matter Expert	The SME knows the business of the customer and will be able to give and validate business requirements for the team.	Scotty knows Paula's business.

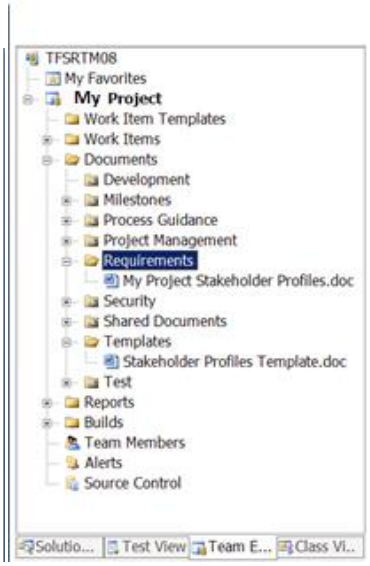


Figure 8: Sample Stakeholder Profile document and storage location

Once the Stakeholder Profile document is complete, save it in the Team Project's "Documents→Requirements" library for the project. In fact, for any details that the team captures relative to the requirements that do not need to be maintained in the tight trace hierarchy (and we have not provided a work item type specific to it), should be documented and placed in this library for the team to share.

Information about stakeholders will help you in preparing and planning requirements elicitation for your project.

3) **Determine Root Cause** – Using the preliminary problem definition and working with the problem owner stakeholder and subject matter experts, you should be able to use elicitation techniques such as "fishbone diagramming" and "Pareto Analysis" (Refer to the "RM Rangers – Requirements Elicitation Guidance" on Elicitation Techniques for more specific guidance on techniques for eliciting requirements) to identify the largest problems to be solved and the mechanisms and scenarios within the organization that cause the problem. This information might also be described in a customer's scope/vision document. If it's not, you'll need to note this for yourself. We have not provided a work item for this information, but there is a document template similar to the Unified Process Vision document that makes a good business summary document for the customer. It should be filled in using a similar method as used to define the stakeholders and then stored in your Team Project's Document→Requirements document library. (see the graphic above for stakeholder identification)
4) **Identify Solution Alternatives** – This step in the process may not be needed if the customer has already decided on a high level solution among several alternatives. If they have not, this step will produce a few solution definitions at a high level that provide the customer with enough information to make an educated decision based on cost, feature richness, technical and business risk, simplicity of enterprise architecture and adoption, etc. For the purposes of most of our

implementation teams, we've determined that this step will not occur frequently enough to go into specific details in this guide. Alternatively, engage or hire a business analyst whose skills are specific to this type of analysis.

5) **Identify the Solution Features** – Business requirements will be captured as Feature work items. Using interviews, requirements reviews, brainstorming, and other elicitation techniques, gathers the business requirements for the customer's decided solution. Refer to the "RM Rangers – Requirements Elicitation Guidance" on Elicitation Techniques for more specific guidance on techniques for eliciting requirements. Be sure to account for the following categories while eliciting business requirements:

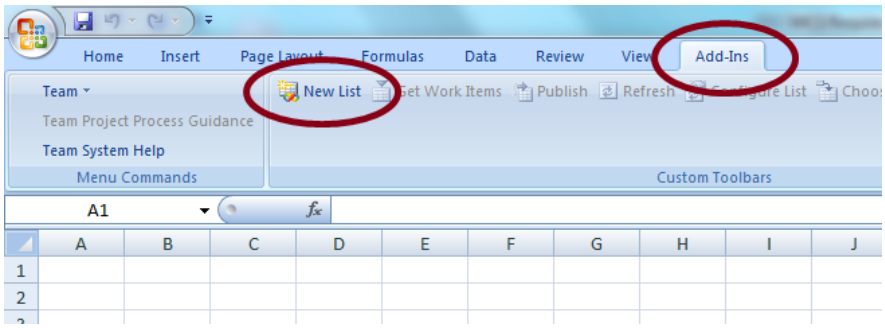
a. Functionality	b. Performance
c. Reliability	d. Support
e. Operations	f. Administration
g. Security	h. Database Administration
i. Enterprise Architecture	j. Application Architecture
k. Documentation	l. User's Guides / Online Help
m. Training	n. Usability (novice, expert users)
o. User Experience (i.e. 5 keystrokes to finish any task)	p. User Accessibility (support for handicapped individuals)
q. Other categories that we have not provided.	

Start with this as a checklist for comprehensively identifying all solution features and constraints.

IMPORTANT: Do not just settle on this list. In your work with your customer, you may identify additional categories. Capture them and build your own checklist for future endeavors.

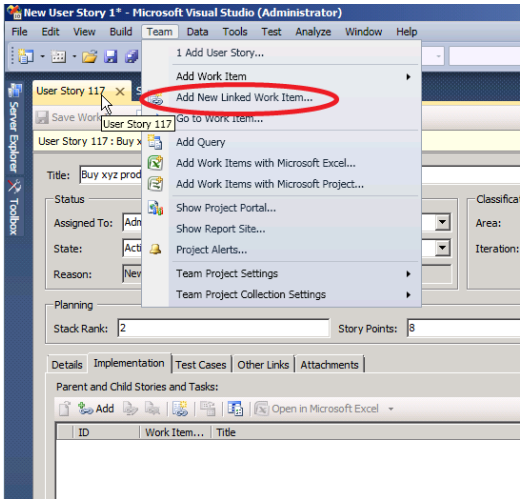
Document each of the features as "Feature" work Items in Team Foundation Server. To simplify the effort, begin with a spreadsheet, connect to Team Foundation Server as a datasource, select the "All Features" work item query as the template, and enter your work items in the spreadsheet.

When completed, simply publish the work items to Team Foundation Server using the publish function on the toolbar ribbon of MS Excel.



Feature Validation

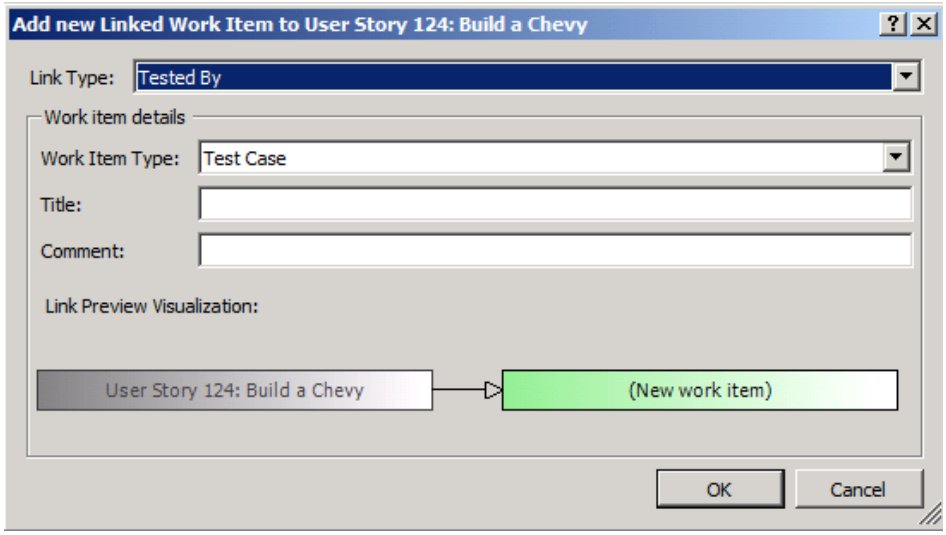
1) **Validate the Feature Requirements** – For each of the requirements captured, ask your customer the simple question: "When I give this to you, what will tell you that I've been successful?" The customer may not know how to answer the question, so you and your team may need to offer suggestions and make them quantifiable. When the answer is solidified, enter the result as a Test Case work.



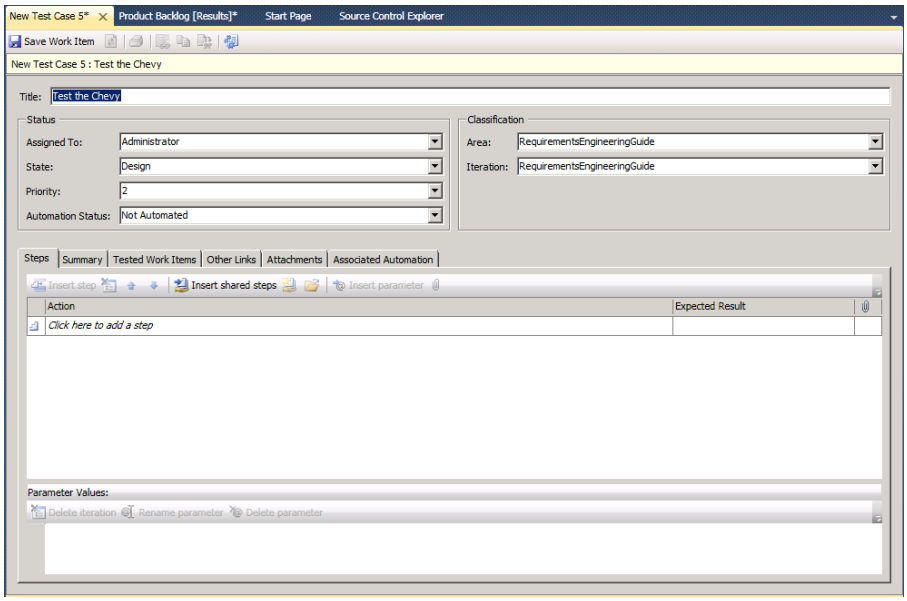
The reason we want to capture the acceptance tests in a vehicle other than the feature is so any work allocated to specifying and implementing them can be tracked.

2) **Create Related User Acceptance Test** – Select the feature that you want to create the test case for, right-click, select "Add New Linked Work Item…". Select a Work Item Type of "Test Case" and

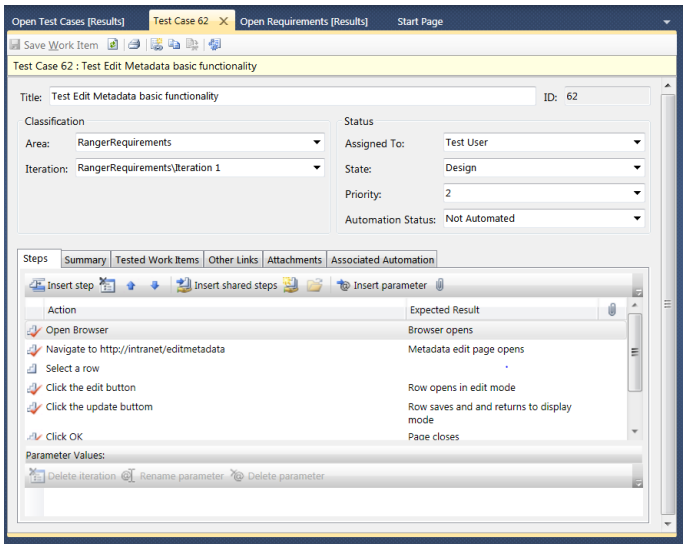
a Link Type of “Tested By”.



This will open a form for entering the test case information and will link the resulting work items to one another.

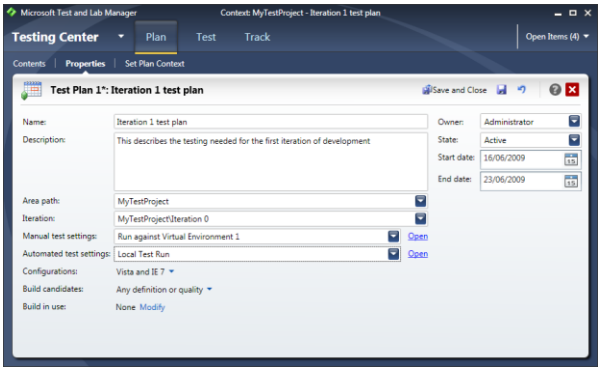


Microsoft Test and Lab Manager (MTLM) is new to the Visual Studio suite of applications in 2010. MTLM provides generalist functional testers the ability to view requirements and to create test cases to adequately test the requirements. Test cases can either be created in MTLM or in Visual Studio. Test cases are made up of steps which need to be performed during test execution.



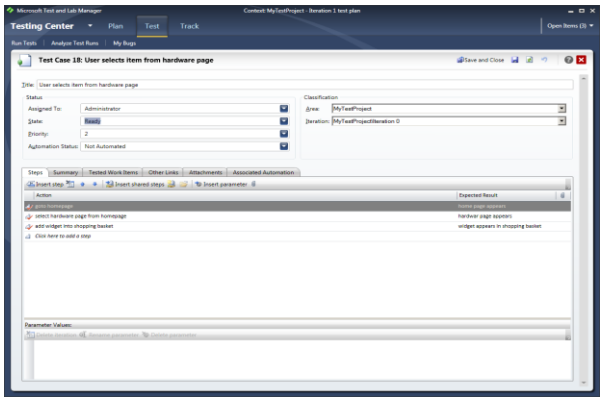
When a test case is created from a requirement it becomes a related item in the Test Cases of the requirement.

Test Plan configuration



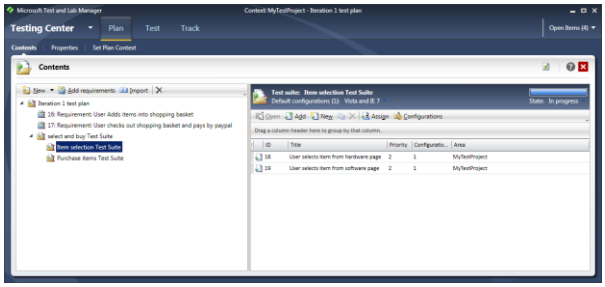
- **Create Test Plans. Why? Help determine the testing effort involved and organizing the test team's activities to ensure maximum test coverage.**
  - Create a Test Plan and set the properties for the plan that include adding default configurations and default test settings
  - Add requirements or user stories to be covered into the plan, link these with Test Suites and add Test Cases and assign tests to testers
  - Report on Test Plan progress.

Test Case configuration



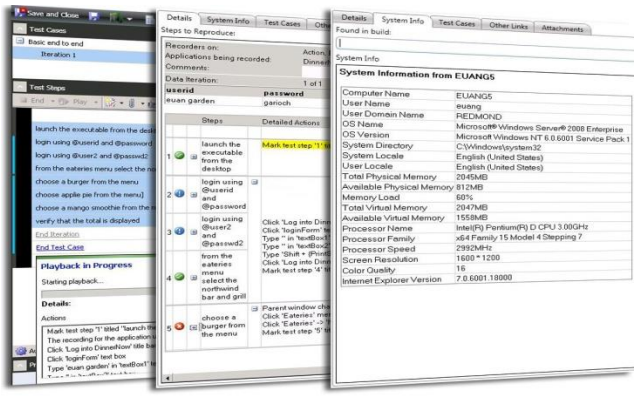
- **Manage Test Cases. Why? Allows for the definition of Test Cases in a clear and consistent way. Test Cases can be linked to user stories, tasks, bugs and test automation which enable easy tracking of the current state of test Cases.**
  - Create Test cases in Team Explorer. Link them to relevant requirements, user stories, tasks, bugs and tests and group into Test Suites
  - Assign to testers, and execute the tests associated with the test cases recording results.
  - Create reports on test case readiness.

Test Suites



- **Create Test Suites. Why? Makes it easy to manage large numbers of Test Cases.**
  - Decide what Test Suites you want to create and how they are related hierarchically. E.g. by functional area or tester.
  - Add the relevant Test Cases into the Test Suites.
  - Manage Test Suites within your Test Plans.

Microsoft Test and Lab Manager



- Lab Management easily tests a variety of configurations in a virtual lab environment, and help developers more easily reproduce bugs by delivering snapshots of environments via virtualization.
- Automated User Interface Testing with Coded UI Test with new Record and Playback Engine

Functional Level Analysis

Functional analysis begins with an accepted set of product requirements (or individual in the case of an agile project) in the form of business or feature requirements. These requirements will be analyzed with the user stakeholders to determine the functions that the users will achieve with the product. It is important that the list of stakeholders already identified is reviewed to ensure that all user goals are captured. Examples of missed user requirements are administrative functions like security administration functions and functional data conversion or installation. If the administrator is not identified as a stakeholder, these projects will end up trying to get production data in place at the 11th hour of a project just to allow the production environment to be established.

The Process of functional analysis will enable the evolution of the specificity of the manageable requirements for a software development project.

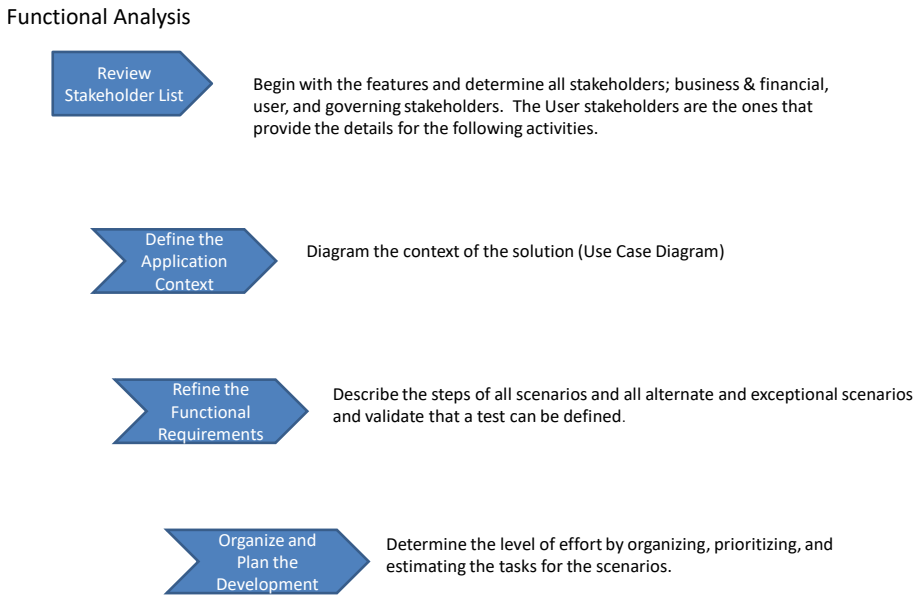


Figure 9: Functional Level Analysis

1) **Review stakeholder list** – as stated above, it is important to identify the common user profiles as well as the mundane in order to be in a position to comprehensively identify and categorize all of the application's functionality. Review which stakeholders are going to be appropriate to identify the user functionality of the system and any integration to other systems that the application will require (SME's).
2) **Define the application context** – Also known as context diagramming or use case modeling, this activity takes a functional look at all of the goals of the user stakeholders and codifies them as functional requirements. These requirements are best described "Requirement" work items whose "Type" attribute is specified as "Functional". These requirements will provide the basis for technical analysis (covered in the next section) as well as project estimation and planning. The deliverables of this activity are typically defined by requirements, identified as work items in Team Foundation Server, as well as a context diagram or use case diagram that traces to each of the requirements. If drawing the context or use case document in Visio, Word, or an alternative, those documents should be stored in the Team Project's Documents→Requirements" document library.

   The 'bubbles' or names of functions on the diagram provide the titles of the functional requirement work items. At this point, it is not necessary to have all of the detail of the functional requirement's work flow, as it is mainly a placeholder for that future work which could be delayed until a separate iteration or increment of the application project.

3) **Refine the functional requirements** – The functional requirements are not usable until their details are specified. So, if the requirement has been identified as the target for a period or increment of the project (iteration), this is the time to specify those details. The procedures followed to execute the functional scenarios need to be described in enough detail to develop the application as well as describe the test steps.

   Specify this information in the description field of the Requirement work item. If it becomes overly long, consider breaking the work item into multiple work items or specifying the detail in a functional specification; one per work item. Not one large specification that describes all of your work items. That type of specification is not that usable by a collaborating team.

4) **Organize and Plan the Development** – Using the functional requirements as the starting point, project managers will need estimates for the effort going forward. There are still unknowns due to the incomplete non-functional analysis (Technical Analysis), but there is enough information to begin the planning, design, and development effort. The results of this activity are tasks and test cases that describe the effort required to accomplish the development of the application solution.

   Using the same procedure described above for identifying user acceptance tests related to features, identify "Task" work items as related work items to each of the functional requirements.

   For the details of the tasks, provide the following:

   - Task Title – high level name or phrase that describes of what the effort will be
   - Task Type – Management, development, testing, infrastructure, documentation, etc…
   - Original Estimate – a number in hours that you believe it will take to complete the task.
   - Work Completed – Zero (0) hours
   - Work Remaining – this should be a number (in hours) equal to the estimate. As the project progresses, this number will change, as will the work completed, but the estimate needs to remain constant so the project team can learn to estimate more accurately and discover their capacity to deliver work.

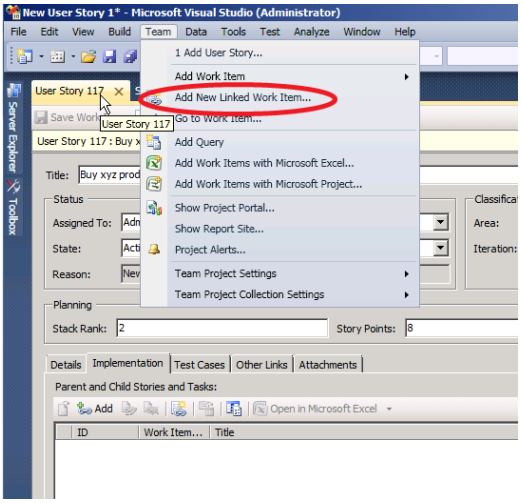
Functional analysis culminates with enough data to begin design and development activity. In a traditional project, this means formal documentation that describes the steps and screen shots of each use case identified. For an agile project, it will be a user goal and its functional test steps at a high level. Either way, functional analysis involves identifying all user and system actors that will participate in roles during the usage of the product. It is typical to develop context diagrams or use case models as a result of this activity.

During this stage of analysis and breakdown, the analyst should capture rough estimates of the work required to implement the function. They should identify an architect and a development lead to assist in estimating the work. Another typical outcome of this activity is a set of tasks (procedure described

above) that represent all of the design, development, and test activity required to implement each function.

Functional Requirements Validation

1) **Validate the Functional Requirements** – For each of the requirements captured, ask your stakeholders the simple question: "When I give this to you function, what will tell you that I've been successful?" The stakeholder may not know how to answer the question, so you and your team may need to offer suggestions and make them quantifiable. When the answer is solidified, enter the result as a "Test Case".



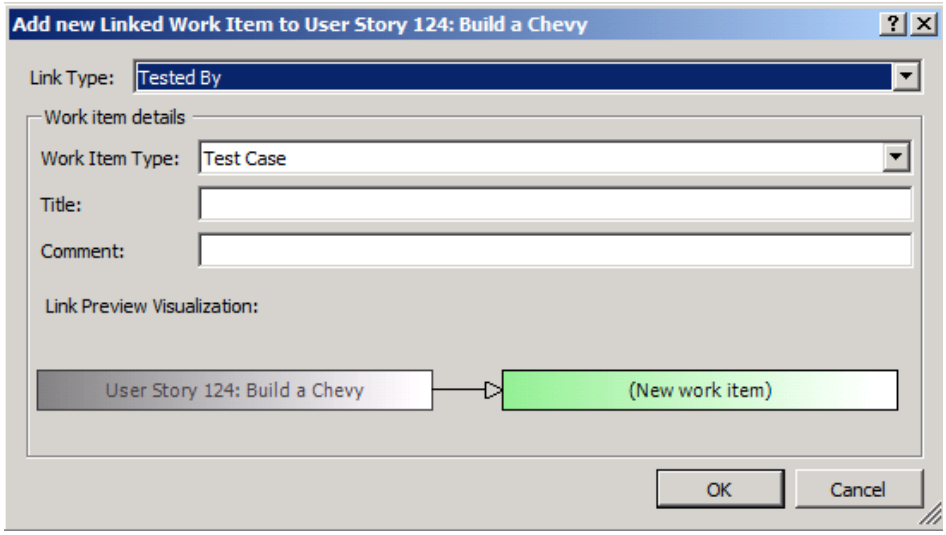
The reason we want to capture the system tests in a vehicle other than the requirement is so any work allocated to specifying and implementing them can be tracked.

There will be, by necessity, more than one test identified for each requirement. You will want to brainstorm positive test cases, negative test cases, exceptions, and all permutations of data that vary for the execution of the function. In addition, you will want to enter any intricate field validations as independent tests.

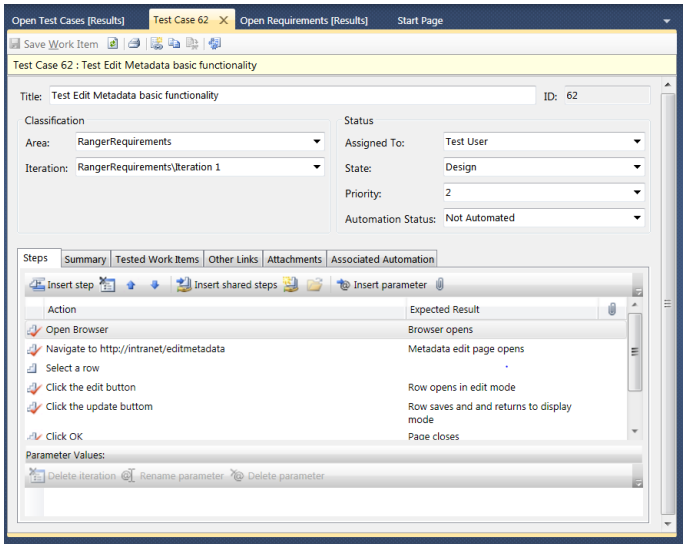
This may seem like an over-abundance of work, but will pay off in the long run resulting in a higher quality application that will not be returned to the development team for rework. The idea is, "If I don't identify the test case, I won't plan the test. If I don't plan the test, I won't execute the test. If I don't execute the test, my customer will identify a defect after I've given them the software. So reduce the re-work headache now.

2) Create Related System Test – Select the Requirement that you want to create the test case for, right-click, select "Add New Linked Work Item…". Select a Work Item Type of "Test Case" and a

Link Type of "Tested By".



This will open a form for entering the test case information and will link the resulting work items to one another.



Provide just a short description of the test procedure and any pertinent data as the detail will be specified using a Visual Studio Manual Test project that will be linked to this work item at check-in time and its results captured and stored in the Team Foundation Server data warehouse during execution.

Technical Level Analysis

Technical analysis begins on the heels of functional analysis. The complete estimate of the product effort cannot be accomplished until the technical requirements are identified. These requirements will

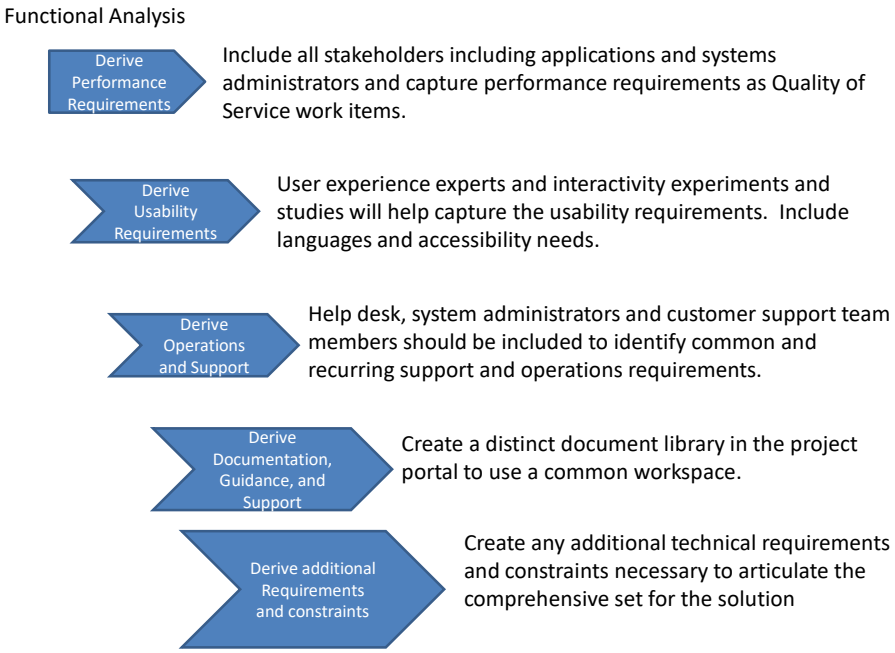
be analyzed with the operational, support, infrastructure, and other non-functional area stakeholders of the application to determine the qualities of service and non-functional requirements of the application.

The process of technical analysis will enable the evolution of the qualities applied to the functional requirements for a software development project. What follows is a description of the high level steps for identifying the technical requirements. The procedure for capturing them as "Requirement" work items is the same as it was for functional requirements and their related tasks, so it will not be described here.

1) **Derive Performance Requirements** – If the organization has service level agreements, begin with these to determine measurable performance requirements. These requirements are going to determine reliability constraints (hours of uptime, days of service without failure, etc…), load requirements (expected numbers of concurrently logged in users, parallel transactions, etc…), scalability (number of transactions per hour/minute/second, number of users executing similar functions, etc…).
2) **Derive Usability Requirements** – Describe the ease of use and access in terms of common look and feel to other applications, ease of self service (user should be able to execute a transaction without any assistance), functionality catered to different skill levels (novice vs. expert user) and profiles (peak time access vs. casual use), etc… User Experience and ergonomic experts can assist with this.
3) **Derive Operations and Support Requirements** – Engage with the organization's operational support or help desk groups to determine support functionality and troubleshooting capabilities.
4) **Derive Documentation, Guidance, and Training Requirements** – Determine the amount of documentation that is required to accompany the application on delivery. User and support stakeholders will be of assistance to elicit these requirements. Creating a separate document library in the Team Project portal site will provide a single location for these artifacts that is easily accessible to all team members as well as provide context for the contents.

   IMPORTANT: Please note that our engagement teams execute based on a signed "Statement of Work" which includes **"Deliverables"** for the engagement. Deliverables should be identified as "Requirement" work items whose type = "Documentation". Tasks for writing the documents should be identified using the same procedure described above, including original estimates, work remaining, and work completed. They should be tracked using the same mechanisms that are used to track the rest of the project requirements.

5) **Derive any additional technical requirements and constraints as necessary** – Not all of the types of requirements can be known at the onset of a project, let alone to describe in a guidance specification for performing requirements analysis. As such, add a 5th step to the technical level analysis below and make it specific to your effort.



Task Analysis and Project Planning

In managing a project, analysis of the requirements in order to identify tasks and their estimates is another form of analysis that contributes to the breakdown of a work plan for implementing those requirements. Visual Studio and Team Foundation Server 2010 provide new capabilities that enhance the project manager's experience at identifying, tracking, and controlling the execution of tasks for a software development project.

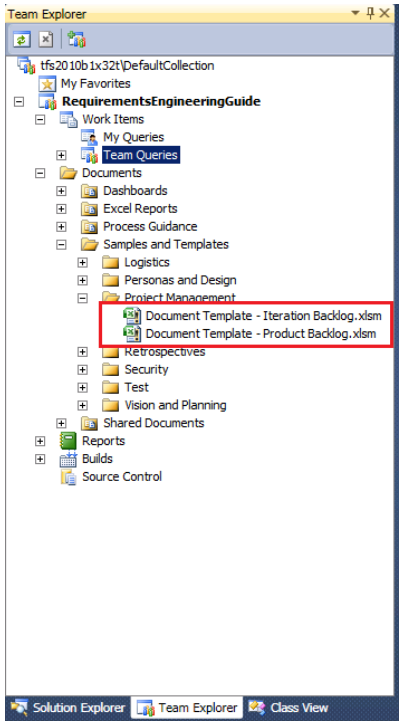
Visual Studio 2010 work item management has been improved since 2008 to provide better project planning capabilities to project managers. One of the most popular project management utilities used in the industry is Microsoft Excel. This is because of its record listings and pivot table capabilities that allow for rich charting and calculations.

Project planning, estimation, and tracking for Visual Studio 2010 using the MSF for Agile process template were designed to leverage Excel for its purpose. Two workbooks are standard templates that come with the MSF for Agile process template. Though it is designed to work with MSF for Agile, there is no reason users couldn't extend them to work MSF for CMMI or any other process template for that matter.

There are also new features in Visual Studio that give better integration with Microsoft Project. Because of the new hierarchical link capabilities provided by the implementation link, traditional project management practices can also be supported in the 2010 edition of the product. Specifically, nested tasks now provide roll-up capabilities on estimates and work remaining that was not possible in the previous versions of this integration.

Here, we demonstrate the agile project planning capabilities provided with the Excel workbooks that are part of the MSF for Agile process template.

One workbook represents the product backlog and the other is the Sprint Backlog. They both consist of an extract of the user story work items and their implementation links; tasks. (Refer to the RM Ranger Hands on Labs for 'point and click' guidance').

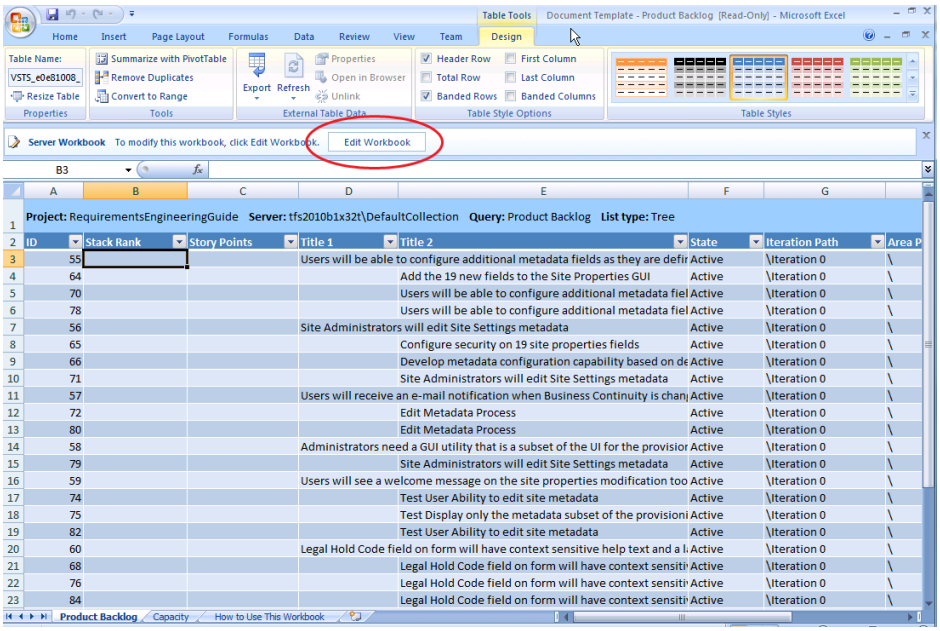


Product Backlog Work Book

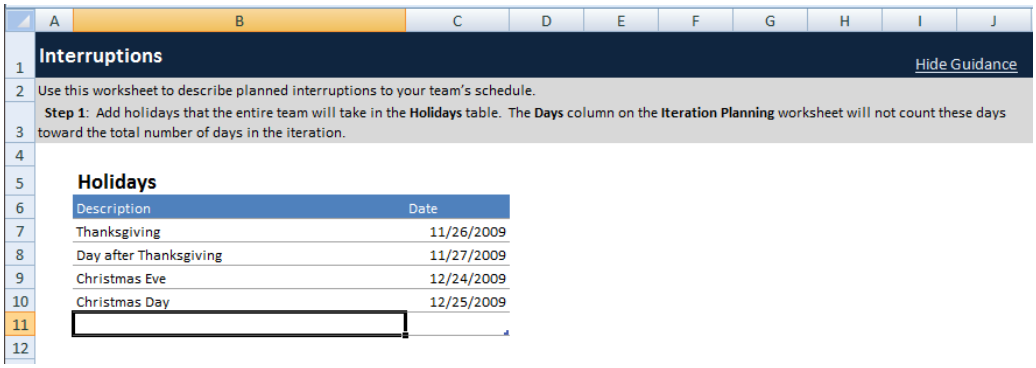
The Product Backlog work book has 3 worksheets.

- Product Backlog – a hierarchical listing of the user stories and their tasks. We will use this sheet to edit the relative sizes of the user stories, the force ranking of them, and the target iteration. This will help in organizing the iterative planning effort. If this was a non-agile project, the same information could be adjusted, but the iterations would need to be a single iteration or organized by major releases. If you were to extend the workbook to reflect MSF for CMMI or another process implementation, make use of the relative sizing and estimates, rank, and priorities to support project visibility.
- Capacity – this worksheet allows you to identify all of the iterations for the project, their dates, number of users, and the amount of work that can be expected during the project. This information will help you gain visibility into over or under-loaded iterations, allowing you to re-distribute the load. If extending for a waterfall project, the iterations could easily support quality gate or methodology milestones.

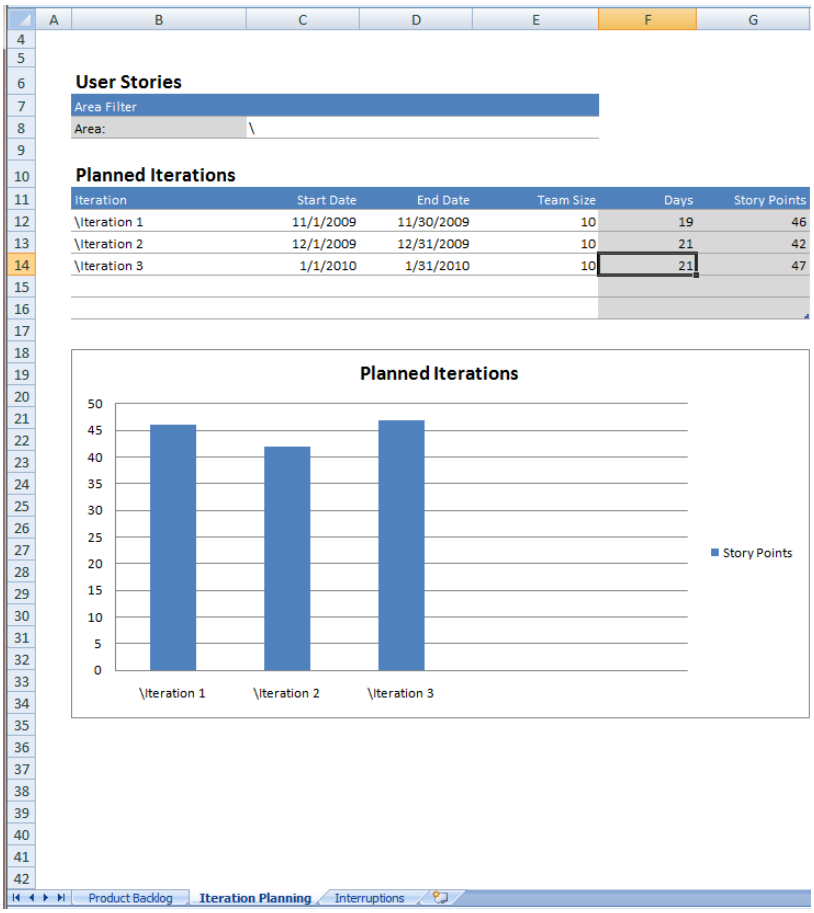
- How to use this workbook – this worksheet is instructional guidance on how to use the worksheet. It has much more information about it than is documented in this paper, so please read it when you have more time.



1. The Interruptions worksheet provides details that are not stored in work items, but leverages that data to derive planning and scheduling information:
    - Enter dates for any Holidays on the work plan



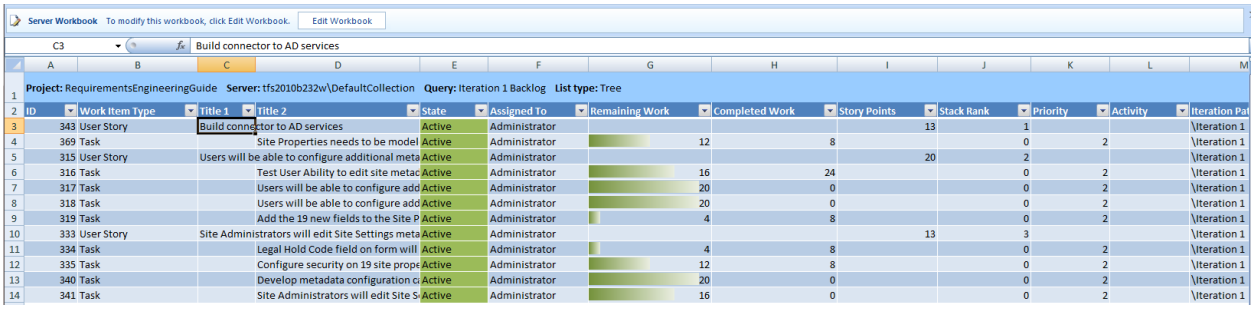
2. The Iteration Planning worksheet provides a distribution of story points (relative sizing) for each milestone. Users will enter the iteration start and end dates and number of resources on the team for its team size. Then, it gets the rest of its data from the sum of the story points value of each User Story targeted for the milestone. It calculates the working days from the system calendar and the identified Holidays from the interruptions worksheet. It will also use all of that information to graph the project resource capacity for each milestone. Basically, it calculates how much work is planned based on the number of requirements and their relative sizes (story points).



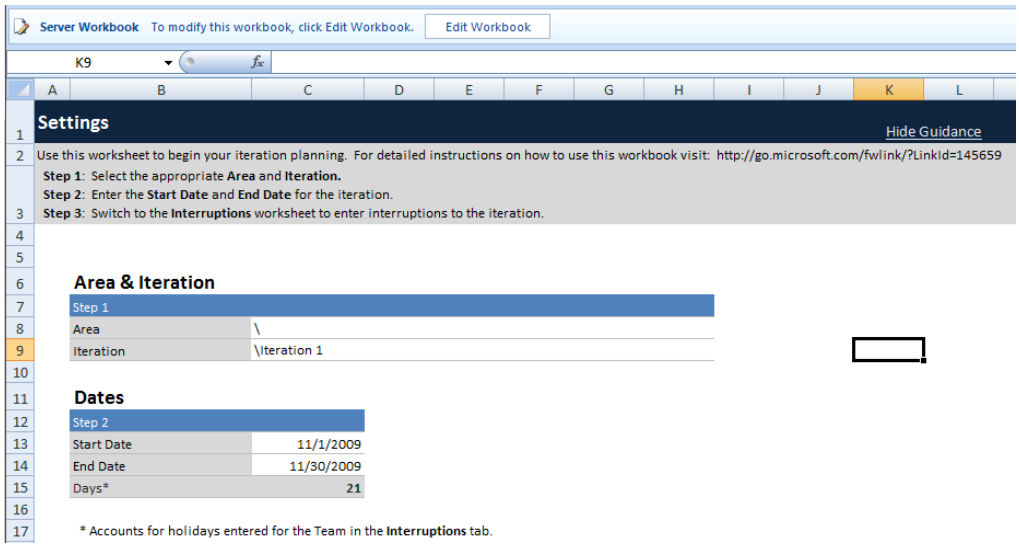
Sprint Backlog Work Book

The Sprint Backlog work book has 3 worksheets.

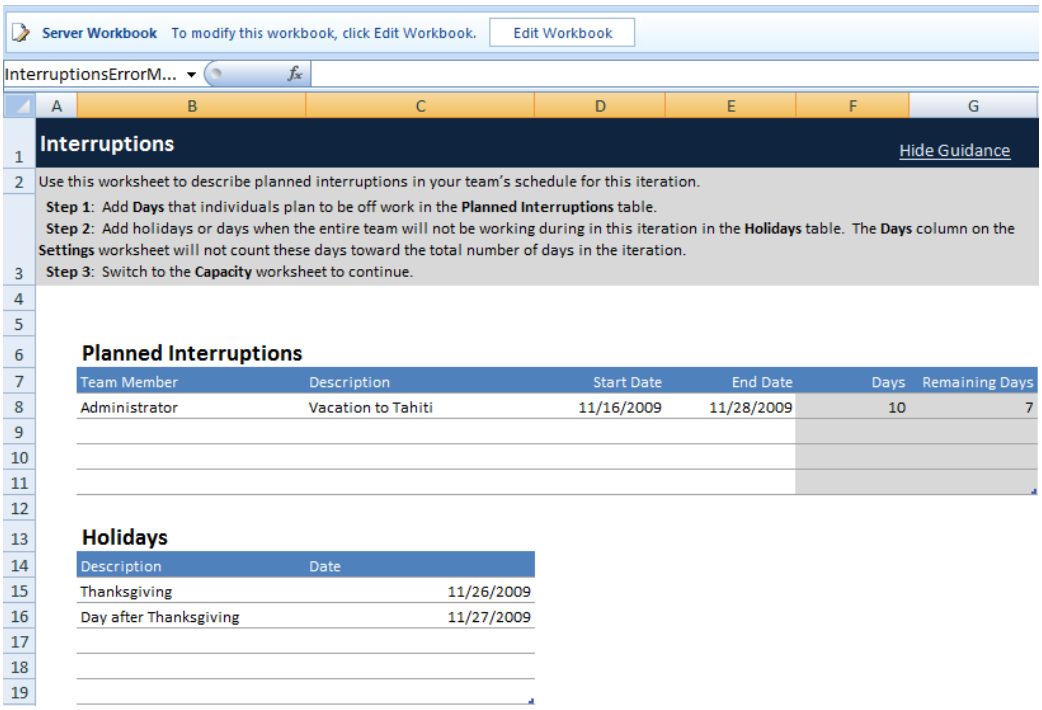
- **Iteration Backlog** – a hierarchical listing of the user stories and their tasks. This sheet allows us to manipulate which stories and tasks are assigned to which iterations. Refresh or publish the data when needed. Also, this list should be filtered on the iteration that the team is planning and managing, thus focusing the view on the current data. In this view, additional tasks can be created using the Implementation link (or parent-child hierarchy) and original estimates, work remaining, and work completed can be entered. This data is not only synced with Team Foundation Server, but it also drives the other worksheets in the workbook.



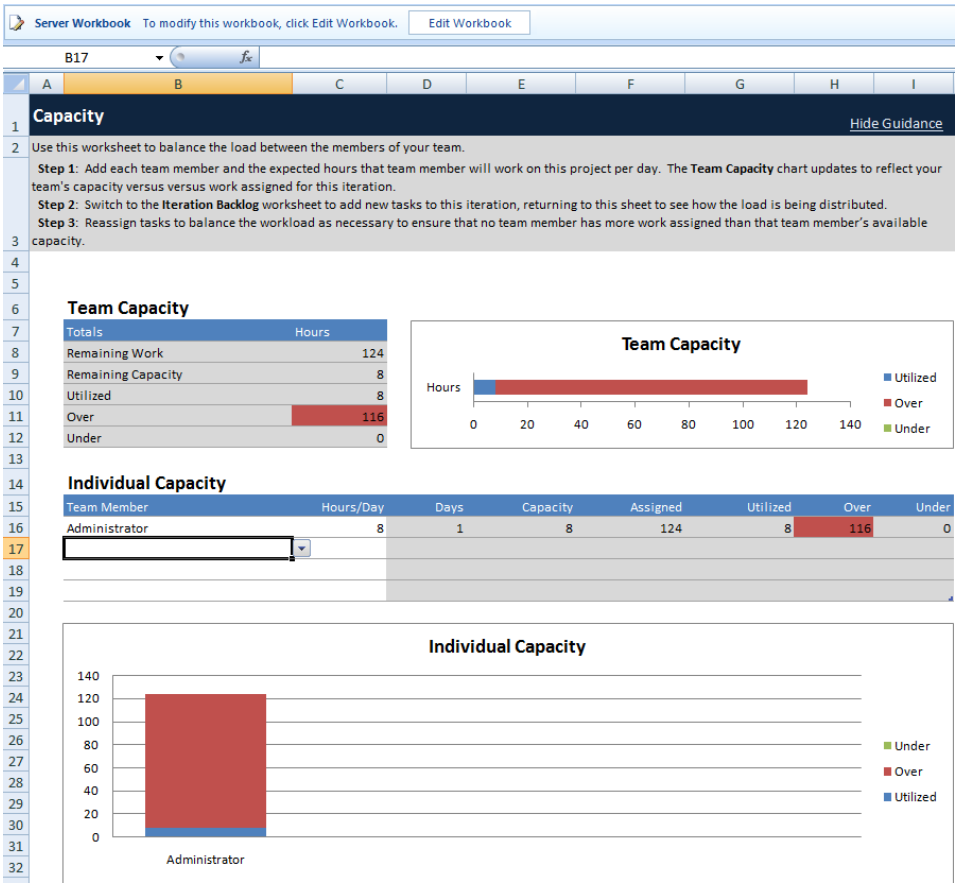
- **Settings** – this worksheet provides a mechanism for establishing the area and iteration for the worksheet data and the start and end dates for the iteration



- **Interruptions** – this worksheet, as with the product backlog workbook, is for entering holidays and other day long interruptions to the project. It is also for entering interruptions for individual members of the development team

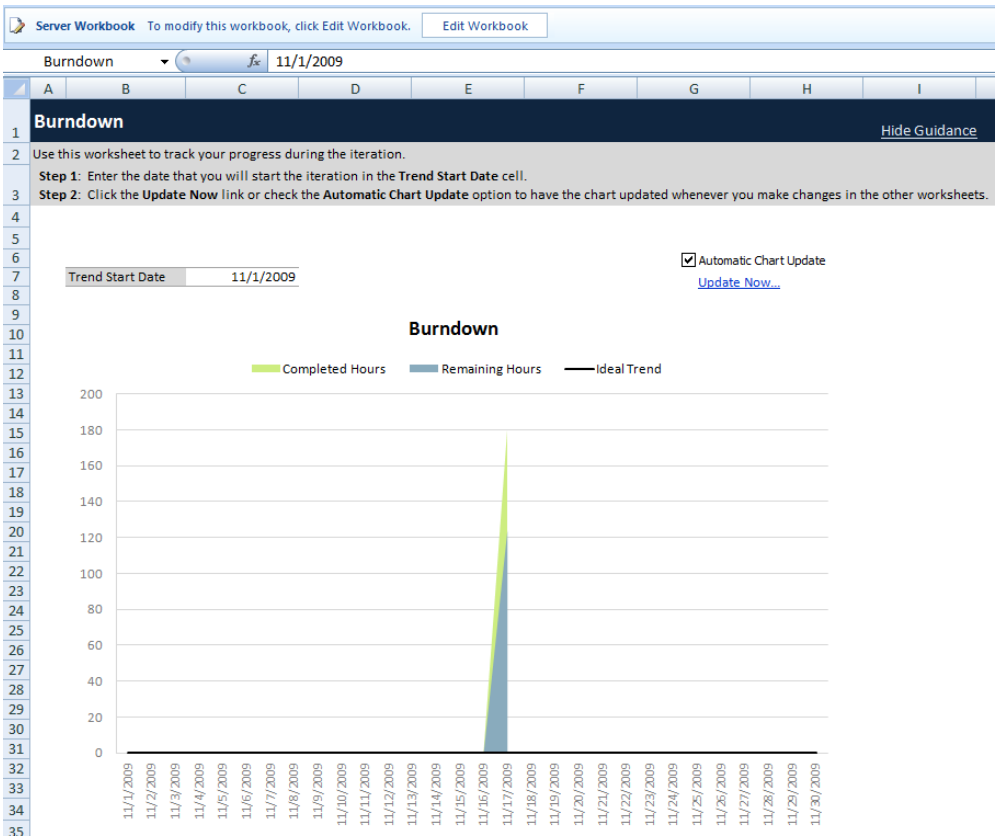


- **Capacity** – this worksheet allows you balance the load among team members.



When kept current throughout a project, this data can give a project manager visibility into over or under utilization… allowing them to make mid-iteration adjustments.

- **Burndown** – This worksheet represents a time series chart for the chosen iteration that lets you know progress made for each day in the period and maps it against a trend line that lets you know if the iteration is trending toward completing on time or not.



It is easy to see that this information, when kept current throughout a project, can give a project manager visibility into trends indicating that they will complete on time or not… allowing them to make mid-iteration or project adjustments.

Final Thoughts on Analysis and Breakdown

Analysis and breakdown is about evolving the requirements to enable higher quality of the application that is delivered by the team. In doing so, the team should employ the techniques described in the elicitation and validation topics to ensure appropriate and comprehensive allocation of requirements for the project.

Analysis techniques include workshops, inspections, acceptance checklists, and test plans.

Storyboarding and activity diagramming are good techniques for describing functional requirements. Refer to the elicitation topic area for more guidance.

Requirements Elicitation

The following extract was written in 1992, 17 years earlier than this Visual Studio ALM Ranger exercise to develop requirements engineering guidance:

> [1]"Many of the problems in creating and refining a system can be traced back to elicitation issues. Yet, much of the research conducted on requirements engineering has ignored elicitation, leading Leite (the author) to claim in a 1987 survey on requirements analysis that the state of the art in requirements analysis is not much different than it was in the late 1970s. He argues that there is a concentration of research in this area on precision, representations, modeling techniques, verification, and proofs as opposed to improving the elicitation of requirements. He concludes that 'research efforts should be directed towards methods and tools needed to improve the requirements analysis process, and, in particular, to those providing more support to the elicitation of requirements'."

That said, here it is 2009 and requirements elicitation is still a neglected practice. This topic area will describe planning, elicitation techniques, and resulting storage support that Visual Studio and Team Foundation Server provide for Requirements Elicitation on application development projects.

Requirements elicitation is the process of identifying needs and bridging the disparities among the involved communities for the purpose of defining and distilling requirements to meet the constraints of these communities.[2] The process, itself, is not a onetime occurrence on any project. It is an iterative event that consists of the identification of the source of requirements, planning the gathering activities for each source, early requirements gathering, and the further evolution of details from which a solution can be developed that achieves the stakeholders' goals with high quality.

This document will describe the elicitation of requirements at each of its evolutionary levels within the parameters of two different methodology types; traditional application development and agile methods. Visual Studio and Team Foundation Server provide the common technology between both methodology types and serves as the planning, storage, tracking, and reporting repository for all elicitation activity.

> **Note:** The guidance for requirements elicitation found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

[1] Christel-Kang, 92, p. 4
[2] SEI 91, p.26

Generic Elicitation Topics

Elicitation Planning

Elicitation requires a concerted planning effort, especially for large, multi-business application initiatives. As such a plan needs to be more than just a task on a work break-down that allocates a short period of time for gathering requirements.

Elicitation Planning is achieved at 3 core levels, early information gathering, requirements expression and analysis, and validation.

Information gathering involves identifying all relevant parties and gathering initial "wish lists" of requirements from them. Through prioritization based on political, economical, environment, etc… factors, these requirements lend themselves to the next iteration of requirements elicitation.

Requirements expression and analysis involves the analysis of all users' needs for consistency and feasibility. Techniques will focus on functionality in terms of scenarios of user goals as well as non-functional requirements, such as performance, reliability, usability, and support. We'll cover this topic in the section titled ***"Requirements Analysis and Breakdown"***.

Validation, then, will provide a mechanism for ensuring that agreement can be made. In the topic area for ***"Requirements Validation"***, we'll cover testability and gaining agreement by using checklists with end user stakeholders and other internal non-functional stakeholders.

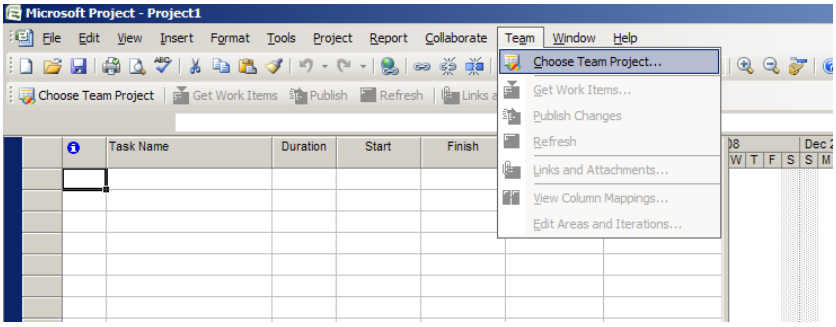
With that, elicitation planning involves the following activities early in a project and then, through refinement, will include activities for further analysis and validation:

1) Identify all relevant parties which are the sources of requirements.
   This activity and the next determine the application's Personas. A critical element of the Microsoft Solution Framework (MSF). The party might be an end user, an interfacing system, or environmental factors. Though there will be roles identified within any organization that are not typical, this is a typical list of party roles:
   a. Business Stakeholder – the person or people providing the funding for the project
   b. User Stakeholder – the person or people that will ultimately be using the system
   c. Administrator Users – in a sense, a user, but special in that they provide security, data feeds, and new user, product, pricing or other information in applications.
   d. Infrastructure – IT organizations that will provide hardware and networking support for whatever application will be hosted. In terms of ISV's or site hosting companies, like SalesForce.Com, they are the people that need to determine the hardware sizing and throughput requirements to support the end users that will use applications hosted at SalesForce.Com sites.
   e. Operations – people or groups that will provide operational support to the end users. They keep the hardware and software running during all usage periods. They also provide help desk support for users that run into periodic defects or outages.

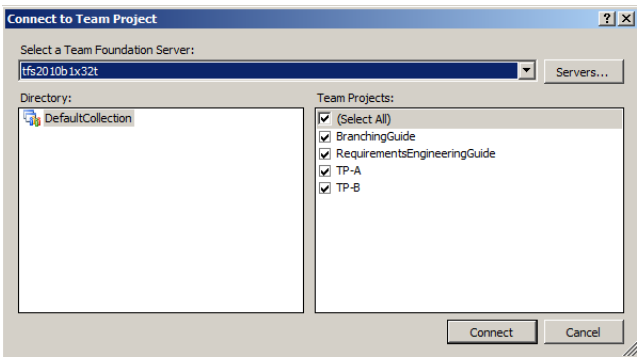
2) Determine the **profile** of the relevant parties to which requirements will be gathered.
   This includes determining the effort required to work with them to get the requirements and the risk of those requirements being accurate and unambiguous. For example, a typical end user in an insurance claims organization will describe their needs in terms of the area immediately surrounding their work space. Screen shots represent pieces of paper to them and when they are submitted, rarely does the user care where it goes, only that they need to know that it has been done correctly. Contrast this to the Chief Information Officer that has given budget to an application development project that is designed to improve the efficiency of the insurance claims processor. Their needs are for low cost of implementation, productivity gains in terms of money and time saved, as well as accuracy of claim that leads to increased margins.
3) Identify the elicitation techniques (next section).
   The appropriate elicitation technique for each profile will depend on schedule constraints, time constraints, depth of knowledge constraints, probability of accuracy, and other factors. Use these factors to allocate tasks with an estimate of effort to elicit the requirements from each source.

To plan for these three activities, use either a spreadsheet or an MS Project Work plan and synchronize with Task work items in Team Foundation Server. The following steps demonstrate how to do this with MS Project:

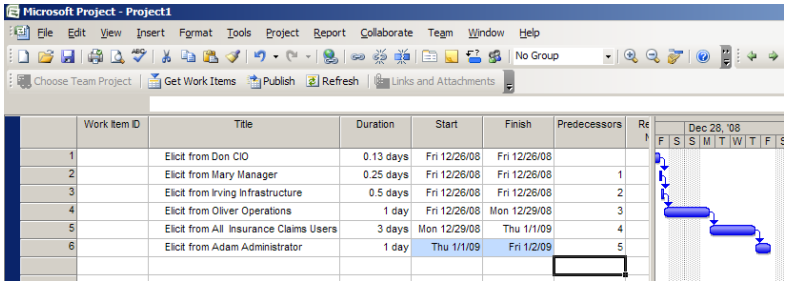
1) Open MS Project and establish the correct attribute mappings by selecting the Team Foundation Server Team Project from the "Team" menu item:



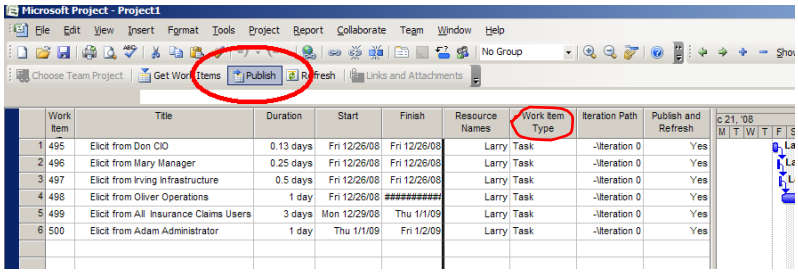
2) Next, select the Team Foundation Server instance and default project collection for which these tasks will be the elicitation planning tasks:



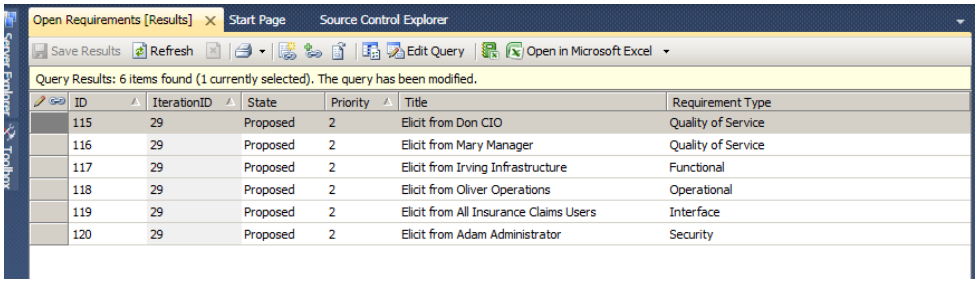
3) Next, enter the elicitation tasks with estimates based on your analysis of each person that represents a persona for your application:



4) To publish successfully, Work Item Type is a required and must be included in the column selection. Once complete, publish the tasks to Team Foundation Server. Make sure to account for the attributes displayed in the screen shot below:



5) This is how it looks in Team Foundation Server once published:



Elicitation Techniques

The techniques described in this section are relevant whether the project is performing in an agile process or a traditional development process. The degree of formality is the only difference that can be applied. At the end of this section, there will be a matrix that describes the difference in each of those contexts.

Elicitation Workshop

A requirements workshop is not so much an elicitation technique as it is a type of meeting or event that organizes several elicitation techniques. It is a good vehicle to start an application development project. It is at the starting point that we are in a position to set expectations with stakeholders and prepare the plan for deeper elicitation techniques that will give us a greater understanding of the problem to be solved and the solution to that problem.

A requirements workshop does not need to be supported by Visual Studio/Team Foundation Server, but its results can be captured in the technology.

We need to capture requests from all stakeholders, and specify how these requests will be addressed. Although the system analyst is responsible for gathering this information, many people will contribute to it: marketing people, end users, and customers -- anyone who is considered to be a stakeholder in the project. In addition to the stakeholders, there will be other sources of requirements that will contribute to the initial elicitation exercise.

Other examples of external sources for requirements are:

- Statement of work
- Request for proposal
- Mission statement
- Problem statement
- Business rules
- Laws and regulations
- Legacy systems
- Business models

This information will be organized, reviewed, and discussed during a requirements workshop.

A requirements workshop is one of the most cost-effective and time-efficient means to get feedback. The same concepts are used in JAD (Joint Application Development) or RAD (Rapid Application Development) sessions. Here are some benefits of a workshop:

- Accelerate the Elicitation Process
- Gathers all stakeholders together for an intensive, focused period
- Facilitated to ensure focus and progress
- Vehicle for hearing all voices
- Results immediately available
- Provides a framework for applying the other elicitation techniques

Workshop Mechanics:

Typically, on a large project, a workshop can last 3-5 business days. Here is a high level procedure for executing a requirements workshop.

1) **Pre-Workshop Planning** – Prior to the workshop, the business analyst role will prepare all materials, identify all stakeholders, send invites to the participants, prepare a topic agenda, and prepare the rules and course of action for the workshop. Here is a list of activities that need to be accomplished:
    a. Sell the workshop – Some stakeholders may not want the workshop for various reasons. Prepare to position the benefits for each stakeholder that addresses their needs.
    b. Establish the team – Identify all stakeholders and facilitators. Prepare to clear their schedules for the workshop.
    c. Handle Logistics – Reserve conference rooms and break out areas. Plan on ordering meals, drinks, and snacks. Ensure that technology is available; internet, projectors, white boards, easels/pads, index cards, sticky notes, markers, etc.
    d. Send out pre-read and pre-requisite materials – The stakeholders should have all available information as well as the time to review it. Send it to them with explicit instructions of how to prepare for the workshop.
    e. Prepare an Agenda – using time boxes (usually half-day increments separated by lunch with two 15 minute breaks during each session), create an agenda for each of the workshops. For example, as the result of an ALM Assessment, we created a workshop consisting of the following:
        - ***Kickoff*** *→ Monday: 8:00-12:00 – Introductions to all stakeholders, their roles, and an open discussion of their desired goals for the workshop.*
        - ***Session 1*** *– Requirements Engineering Discipline → Monday: 1:00-5:00 – 30 minute introduction to the problems with RE identified during the assessment, 1 hour facilitated discussion using fish-bone diagramming and "why reasoning", 15-30 minute prioritization of the problem, 1 hour brainstorming on the solution, 15-30 minute prioritizing and narrowing down the results.*
        - ***Session 2*** *– Configuration & Change Management Discipline → Tuesday: 8:00-12:00 – same format as workshop 1*

- ***Session 3*** *– Agile Project Management and Monitoring & Control Discipline → Tuesday: 1:00-5:00 – Format is more of a "Use Case Modeling" discussion. Describe all of the scenarios of the Agile team, identify the roles, identify the actions, identify the work products; pretty much describe the process they will follow and brainstorm on problem areas*
- ***Session 4*** *– Software Reuse / Architecture Discipline → Wednesday: 8:00-12:00, same as workshop 1*
- ***Close out*** *→ Wednesday: 1:00-3:00 – Wrap up preliminary summaries of each session, identify work remaining, and describe follow-on actions.*

2) **Workshop Sessions** – Execute the workshop, sticking to some of the basics:
    a. Facilitate – Either you or someone you identify to keep the sessions moving. This person will not inject their opinions, but will offer up suggestions if the session becomes stagnant, just to get it back on track.
    b. Keep On Track – same as (a).
    c. Record Findings – The facilitator may have a hard time documenting the results of each session. Consider hiring a scribe for this purpose. It will be important that they understand what information needs to be captured and documented. It is not an easy task at times.
    d. Summarize Conclusions – At the end of each session, work with the stakeholders to summarize any findings or decisions that were made. Make sure that all participants leave the session understanding all options that were presented and the actions to be taken as a result.
3) **Production of Results** – Find time to take the summarized results and analyze and organize them.
    a. Synthesize the Findings – Organize the results into a format that can be analyzed.
    b. Condense the Information – After analysis, consolidate the information in a way it can be presented back to the stakeholders for follow-on actions.
4) **Follow-Up**
    a. Present the findings to the customer
    b. Identify next steps and actions to implement the final set of requirements.

Brainstorming

Brainstorming is an elicitation technique that can achieve a large base of ideas in a very short period of time. Essentially, using brainstorming in a workshop setting, you allow all stakeholders (participants) to provide any ideas they feel will support the chosen topic. Though the format may vary from analyst to analyst, the basic idea is to generate a list of "things" that apply to a particular topic and then cluster, organize, and reduce the list to a manageable size.

"Brainstorming means to spend a short amount of time, say 15 minutes, where everyone in the room is allowed to say whatever they feel is important to the project. After that, a facilitator leads the group in organizing and prioritizing the results."[3]

Mechanics of a brainstorming session

1) Start out by clearly stating the objective of the brainstorming session.
2) Generate as many ideas as possible.
3) Let your imagination soar.
4) Do not allow criticism or debate while you are gathering information.
5) Once information is gathered, mutate and combine ideas.

When generating the ideas, there are several mechanisms that can be used:

- Sticky notes and Sharpies – each participant gets a Sharpie and a stack of Sticky Notes. They are free to jot down as many ideas as they can and stick them to a wall. It is the facilitator's role to read each note and cluster them together as best they can. Using a time limit, say, 15 minutes, stop the exercise and then have the rest of the participants assist in clustering the notes on the wall. This is a quick way to get the information gathered and organized.
- White Board or Easel Pads – Either each participant raises their hand and the facilitator writes down their idea on the easel or white board, or the participants are each given a marker and, one-by-one, write on the board in sequence until a time limit is established. This gets difficult to organize after the work is done.

After the ideas are generated, the group will 'prune' ideas that are outrageous and then vote and prioritize those ideas that are best suited to the problem.

Other techniques to reduce the amount of self-stick notes are to:

- Have everyone take a simple vote.
- Let everyone prioritize each idea by category (for example, critical, important, and nice to have), which have assigned weights (for example, 3, 2, 1). The sum of the priorities for each idea will tell you its importance in relation to the other.
- Give every participant $100 in voting money (not real money) and have them buy their ideas in any way they want. This will not only identify the most popular ideas, but also force a priority to all of the ideas that are generated.

Documentation of the Results

Regardless of the mechanism used to perform the idea generation, the results should be documented and stored. Team Foundation Server can be used as a preliminary placeholder for these ideas as "Feature", "User Story", or "Requirement" work item types, depending on which process template you are using; MSF for Agile or MSF for CMMI.

[3] Rat99

The facilitator will add a work item for each approved "sticky" and incorporate the attributes to it, such as a weight (dollars of importance, or importance of high, medium, or low). These features can then be the starting point for further analysis and elicitation techniques.

Interviews

Interviewing is an effective, personal form of elicitation and, probably, the most frequently used technique for eliciting requirements. Being successful at it, though, requires effective preparation including questions, time allocation, and setting the right expectations.

First, you must prepare a list of "context-free" questions that can help you drill into the domain with the interviewee. Context-free questions are questions that are appropriate in any domain and, when delivered in an open-ended format, will get the interviewee to talk about the topic with long, unbiased descriptions of their problems and potential solutions.

Context-free questions are:

- Always appropriate.
- Formulated so that it helps you understand stakeholder perspectives.
- Not biased with solutions knowledge or your opinion of what the solution should be.

Questions should get progressively more specific, but not leading. As an interviewer, you should listen for potential problems, pain points, or difficulties in the descriptions that the interviewee provides. These are areas that require more specific questions that are slightly more context-sensitive, but still not leading.

The general process for performing an interview is:

1) Research the background of the stakeholder or user and the company ahead of time.
2) Review the questions prior to the interview and prepare to anticipate some of the answers in order to drill into the problem behind the problem.
3) Refer to the format during the interview to ensure the right questions are being asked.
4) Summarize the top two or three problems at the end of the interview.
5) Repeat what you learned to confirm your comprehension.

During the interview, practice "active listening". Active Listening is a practice where one "Seeks to understand, not to be understood". With that, here is some guidance:

1) Ask the question
2) Listen to the answer given
3) Repeat the answer to ensure that you have heard it correctly
4) Restate the answer in your own words to demonstrate that you understood what you heard.

This process will earn you credibility. It will demonstrate to your interviewee that you honestly care about their answers and are seeking to understand them. Once you understand their answers, you are in a position to analyze them in order to formulate a solution.

The alternative to active listening is offering up a solution before you understand the problem. This can lead to an entire development team building the wrong solution to a problem and will, ultimately, not be accepted.

Don't build a script that has so many questions that it becomes difficult to establish rapport. If you are running the interview well, you will have a great understanding of the domain and the problems with it. You can then document the results in a template that can be used to begin establishing requirements as work items in Team Foundation Server.

Examples of context-free questions used to find system users and interfaces:

- Who is the customer?
- Who is the user?
- Are their needs different?
- What are their backgrounds, capabilities, environments?

Examples of context-free questions that help you understand business processes:

- What is the problem?
- What is the reason for wanting to solve this problem?
- Are there other reasons for wanting to solve this problem?
- What is the value of a successful solution?
- How do you solve the problem now?
- What is the trade-off between time and value?
- Where else can the solution to this problem be found?

Examples of context-free questions that help you understand requirements on the system or product to be built:

- What problem does this product solve?
- What business problems could this product create?
- What hazards could exist for the user?
- What environment will the product encounter?
- What are your expectations for usability?
- What are your expectations for reliability?
- What performance/precision is required?

Examples of context-free meta-questions:

- Am I asking too many questions?
- Do my questions seem relevant?
- Are you the right person to answer these questions?
- Are your answers requirements?
- Can I ask more questions later?
- Would you be willing to participate in a requirements review?
- Is there anything else I should be asking you?

Examples of non-context-free questions that should be avoided are:

- Leading questions: "You need a larger screen, don't you?"
- Self answering questions: "Are fifty items about right?"
- Controlling statements: "Can we get back to my questions?"
- Too long and too complex: "I have a three part question, ..."

When you formulate a set of questions, you also should consider the following:

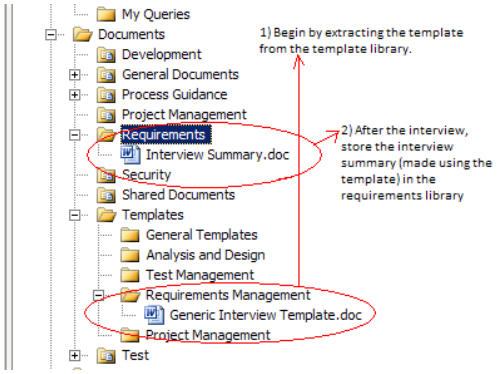
- Don't ask people to describe things they don't usually describe.
- Don't ask questions that assume that users can describe complex tasks. Example: tying your shoelace.
- In general, people can do many things they cannot describe.
- Empirical evidence - poor correlation.
- Ask open-ended questions.
- Avoid questions that begin with "Why?", since such questions can provoke a defensive posture.

When you conduct an interview session, remember:

- Don't expect simple answers.
- Don't rush the interviewee for answers.
- Listen, listen, listen!

Documentation of the Interview Results

The Generic Interview Template referenced at the end of this document in the appendix is a good starting point for preparing an interview. The results can be documented in the template and stored as an Interview Summary in a Requirements Folder of a Team Foundation Server Team Project:



Fishbone Diagrams

Fishbone diagrams provide a mechanism to an analyst performing an interview to drill into the problem behind the problem, or, rather, the root cause of the problem.

A fishbone diagram can be described exactly like a fish's bones after the meat has been eaten. The "head" and backbone of the fish represents the description of the initial problem. Each spine off of the

backbone represents a reason or root cause to that problem. An initial set of "spines" are then analyzed for importance and relevance to the problem. The top 20% of the reasons, then, are analyzed as their own "fishbone" diagrams until the problem is sufficiently analyzed.

It is a good technique because it gets the stakeholder involved in identifying their own root cause of their problem.

Storyboarding, Wireframes, and Prototype Development

In movies, cartoons, and any animation, a storyboard helps the authors describe the story quickly with visualizations that help a team of story tellers or stakeholders understand the content without a lot of effort. Storyboarding can be used in a similar way to demonstrate the functionality of an application in a rapid, low effort way. Some of the benefits of storyboarding are:

- To help gather and refine customer requirements in a user-friendly way.
- To encourage more creative and innovative design solutions.
- To encourage team review and prevent features no one wants.
- To ensure that features are implemented in an accessible and intuitive way.
- To ease the interviewing process – it jumpstarts a discussion when an interviewee is not that creative or talkative.

Another way of describing a storyboard is to say that it allows you to have a tool to illustrate or animate the functionality and behaviors of a system in the environment for which it will run. The development of a storyboard happens in real time with the participants, as it is a changeable artifact that does not require a lot of polish. In fact, if it is cumbersome to enhance, then the act of storyboarding has gone awry.

How to document a storyboard

Storyboards can be low-tech in terms of text and still pictures or it can be animated and quite intricate in the form of a prototype using HTML pages in wire-frames.

Here are some examples:

- Paper sketches or pictures
- Bitmaps from a drawing tool
- Index cards
- PowerPoint slides
- Screen shots (if a user-interface, or prototype of the user interface exists)
  Note: Storyboards expressed in terms of actual screen shots can be a useful input to the end-user documentation.

A Word about Wireframes and Prototypes

Wireframes and Prototypes are another mechanism for describing storyboards. The difference, however, is that they are functional in nature and give both the benefit of a clearer understanding of the implementation and the risk of implying too much of the application is complete. Expectations must be

set with stakeholders that a prototype or wireframe are just samples of the possible solution to entice early feedback, but are not meant to be fully functional. There is still considerable effort remaining.

Examples for possible formats are:

- Expression Blend Sketchflow projects
- Horizontal prototypes written using Web, WPF or Windows Forms solutions
- PowerPoint slides with additional animations or navigation logic

Regardless of format, it is important that the storyboards find their way into your central repository.

If done on paper using hand-drawn sketches, then the images should be scanned as jpg or bmp files that can be stored in the document library in the same way as the results of the brainstorming sessions or other requirements workshops. If done on electronic media, then the results should be either in file format already or generated into a file format for the central repository, using the document library again.

Once in the Team Foundation Server library, work items can be linked to them in order to establish and maintain traceability.

Storyboard Conversion to Work Items

Each frame within a story board (or button click on a wireframe or prototype) demonstrates a behavior that the system can demonstrate. As such, they make good user stories or steps within a use case. In either case, each element of the story should be linked in some way to a work item that represents the requirement or task that the development team needs to accomplish in order to fulfill the development and testing effort.

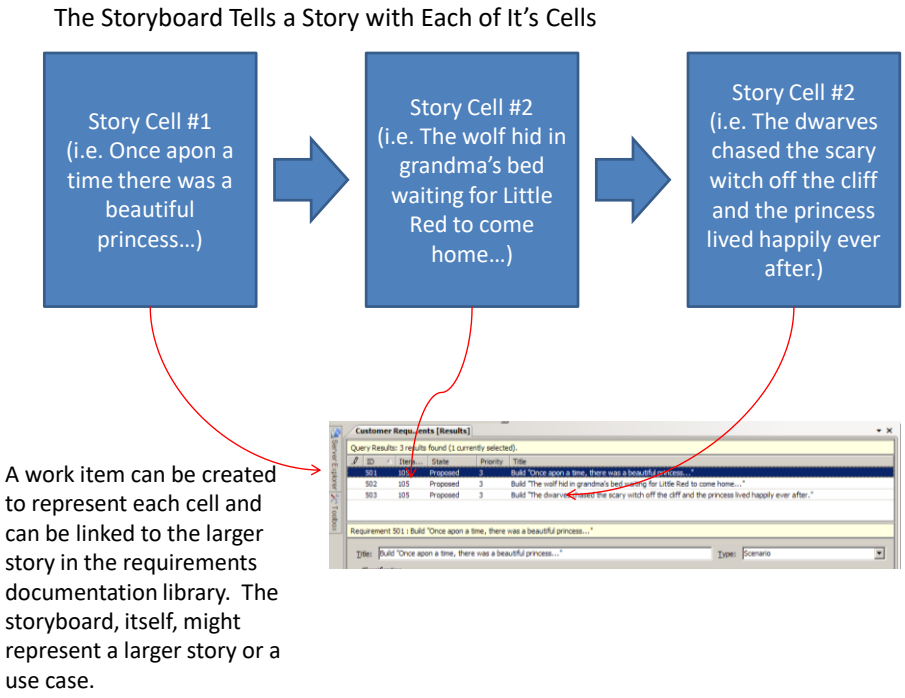


Figure 10: Mapping Storyboard cells to Work Items

A more integrated way to incorporate prototypes and work items is to document use cases with Architect Edition of Visual Studio and to create prototypes using Expression Blend SketchFlow. Use case diagrams contain a visual description of how users interact with your software, and may include links to other artifacts inside your Visual Studio solution. Those so called Artifact elements can be used to attach information, like file references, to the diagram. This is where SketchFlow comes into play.

SketchFlow, which is part of Expression Blend 3, is a tool for creating screen-based prototypes for WPF and Silverlight applications. Solutions created with SketchFlow can be opened and edited inside Visual Studio and the other way round. Figure 11 shows a Visual Studio solution that contains a Modeling project with a use case diagram and SketchFlow project named "UsecasesTestScreens". The latter has been created with Expression Blend SketchFlow before.

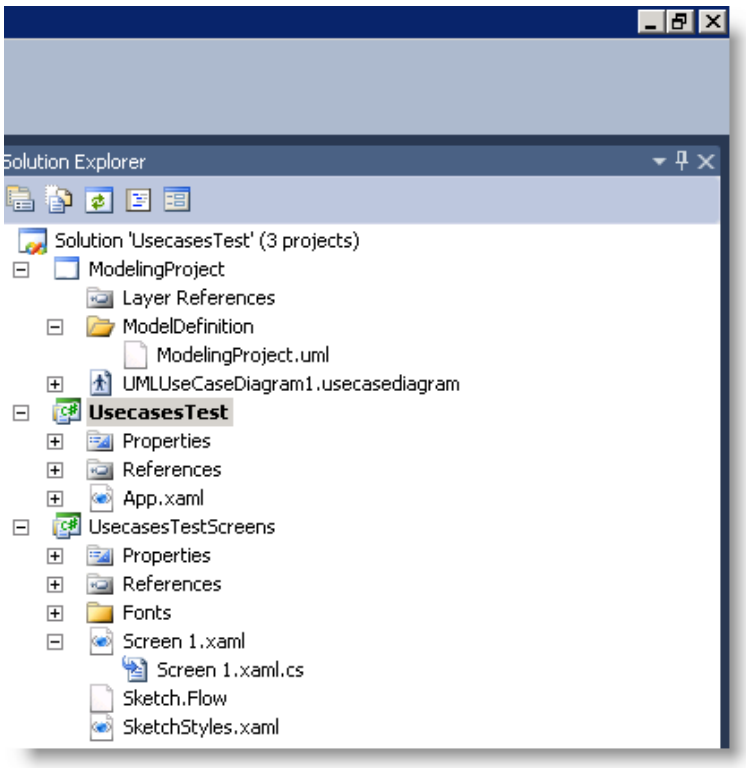


Figure 11: SketchFlow projects can be imported into Visual Studio

After opening a SketchFlow project into Visual Studio and creating a use case diagram within the solution it is possible to simply drag screens, designed in SketchFlow, into the use case diagram. Files dragged into the diagram will be automatically converted to artifact elements and can be used like any other element inside a use case diagram (see Figure 12).

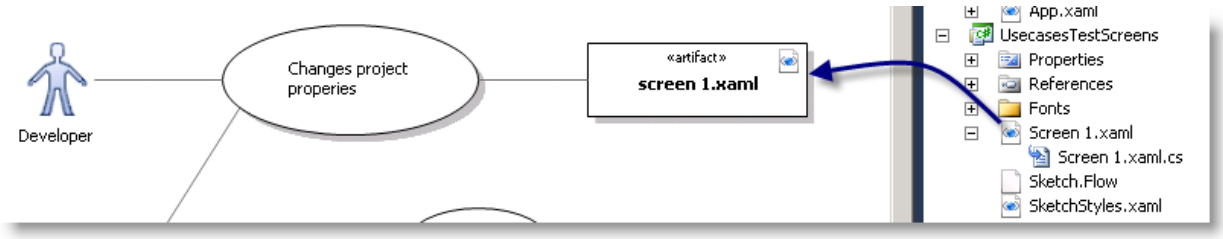


Figure 12: Dragging an UI screen into the use case diagram converts it into an artifact

Double clicking on the screen-artifact will open the associated Xaml file with your chosen xml-viewer, e.g. Internet Explorer. This way it is possible to directly link the use case diagrams with the UI, giving developers a better insight into the project and incorporating the ideas of UI designers at the same time.

Observation

Because many application users find it hard to describe their job, often times we need to assess their work habits and procedures in order to determine better, more efficient mechanisms to help them do their jobs. To do this, observation is a great mechanism used to understand a user stakeholder's role and to identify and ask questions concerning inefficiencies that they themselves may take for granted.

The result of an observation might be a business activity diagram or a sequence of steps representing work procedures (use cases). This documentation should be stored in the requirements documentation library of Team Foundation Server for use during analysis to identify more meaningful functional requirements. Refer to the topic on ***"Analysis and Breakdown"*** for more specific guidance.

Existing Requirements Reviews

Most development projects are kicked off due to preliminary documentation being reviewed and approved. In other cases, existing documentation from completed projects serves as the starting point for starting and enhancement effort. Either way, this documentation should first be organized for the team as a source of record for beginning the project.

Formal reviews of previous documentation can be very beneficial to getting accuracy of the problem to be solved and formulating new requirements for the project.

Mechanics of a Formal Requirements Review

1) Distribute the documentation to be reviewed to each stakeholder for inspection.
2) Guide each person to look at the document from their perspective and document or highlight any issues, additional or missed requirements, or constraints to the feasibility of implementing the requirement. Their perspective is determined by the role they play; i.e. a Test Engineer should strive to define tests for the requirements in the documentation. An infrastructure architect should be mapping new functionality to his/her capacity to support a larger application footprint on the enterprise hardware, or the DBA needs to determine if any of the information will lead to larger storage requirements for the enterprise databases.
3) Use Checklists to validate each document for coverage of each perspective. Refer to the ***"Requirements Validation"*** topic area for guidance.
4) After giving stakeholders time to review independently, bring them all together to discuss their findings. This will give them an opportunity to validate their findings with their peers as well to possibly identify missing requirements.
5) The results of the review should be captured as either requirement or change request work items, or tasks for further analysis or approval.

Agile Elicitation Topics

Most of the practices described above are relevant to all application development requirements elicitation activities. In fact, many of them are derived during events that precede the application development effort. In which case, the agile project methodology does not really apply.

There are some events, though, that are better served to an agile development team by being consolidated and streamlined.

1) **Product Backlog Generation** – For an agile team (Scrum), the product backlog represents all of the requirements that are to be built by the development team. The Product Owner is responsible for building this list. They will use many of the techniques described above to assist in generating the list. The key difference, though, is that they usually don't provide formality to it. The planning effort is scheduled to last a very short period of time and the entire group of stakeholders is treated as a homogenous group of sources for various requirements that become the product backlog. The quality of the backlog is not as pristine as with a traditional development effort. This is neither a risk nor a deficiency, because the agile team will re-engage with the product owner and other stakeholders to drill into the details on a one-at-a-time basis. The product backlog will be stored in Team Foundation Server as Requirement, Scenario, or Quality of Service work items.
2) **Iteration Backlog Analysis** – Once a 'chunk' of the product backlog is approved as the goal for an iteration, the development team members will use mainly interviewing techniques and informal storyboarding to flesh out the details of a user story, relying on the product owner or their delegate to provide the details. The documentation captured is minimal. Just enough to allow the team member to design and map the requirements behaviors to the enterprise/application architecture, write its tests and code, and execute an acceptance test against it.
   For storage, the backlog item is typically a "Requirement" work item in the MSF for CMMI process template, or a "Scenario" in the MSF for Agile process template.

Traditional Development Elicitation Guidance

Most of the detail in the generic section of this topic is redundant for traditional application development. This section will describe only the differences.

In Team Foundation Server, the MSF for CMMI process template provides support for requirements management (ReqM) and requirements development (RD) that an organization can use to assist in their compliance to CMMI Level 3 Capability. Within this template's process guidance, Microsoft has defined "personas" that can be used for determining the users from which to elicit functional requirements.

A persona describes the typical skills, abilities, needs, desires, working habits, tasks, and backgrounds of a particular set of users. A persona is fictional reality, collecting together real data describing the important characteristics of a particular user group in a fictional character. By doing so, it becomes easier to talk and reason about the whole user group, since we can relate to, and understand individuals more easily than we can a group. Each time we refer to a persona, we feel like we are addressing an individual, but we are in effect addressing the group that the persona represents.

An Additional Comment on Traditional Elicitation

Traditional application development is founded on the principles of many regulating bodies such as IEEE, SEI, ISO, and other governing organizations that care about the quality of solution delivery in the software development world. As such, in order to comply with these governing bodies, it is not only important to adopt good elicitation practices described in this paper, but to show evidentiary support that the development team has elicited from the correct groups and made scope, design, and implementation decisions based on these activities.

In order to provide that support, here is a "rule of thumb" for documenting the evidence:

1) If you are to get requirements from a stakeholder:
    a. Plan for the activity and use "Task" work items to estimate and track the elicitation activity
    b. Develop and use checklists specific to the domain and elicitation activity that is to be performed. Store the checklist template in Team Foundation Server in a requirements templates documentation library.
    c. Change the name of the template, fill it in, add your contact information and the date that it was used, then store the result in the Requirements artifacts documentation library in Team Foundation Server.
    d. Link any resulting work items to the artifact in the documentation library.

2) If you are eliciting requirements from previously written documentation:
    a. Again, plan for the activity and use "Task" work items to estimate and track the elicitation activity
    b. Develop and use checklists specific to the domain and elicitation activity that is to be performed. Store the checklist template in Team Foundation Server in a requirements templates documentation library.
    c. Change the name of the template, fill it in, add your contact information and the date that it was used, then store the result in the Requirements artifacts documentation library in Team Foundation Server.
    d. Link any resulting work items to the artifact in the documentation library.
3) If you are summarizing the results of an interview, brainstorming session, or any session within a requirements workshop:
    a. Again, plan for the activity and use "Task" work items to estimate and track the elicitation activity.
    b. Develop and use checklists and templates for documenting the results of the work. Store them in the templates documentation library of the Team Foundation Server Team Project.
    c. Fill in the template, change its name, add the names and roles of all participants, and add the dates of the event to the document before storing it in the requirements artifacts documentation library in the Team Foundation Server Team Project.
    d. Link any resulting work items to the artifact in the documentation library.
4) If you are creating more than just work items to represent a set of requirements:
    a. Develop and use a template that is sharable by the entire organization (Stored in the templates documentation library of the Team Foundation Server Team Project)
    b. Store the resulting document, spreadsheet, drawing, etc… In the Requirements documentation library of the Team Foundation Server Team Project
    c. Link the resulting file to the affected work items as URL Links in Team Foundation Server.

Requirements Specification

Describe the process for entering requirements as work items for each role and work item type.

Guidance regarding requirements specification varies depending on your software engineering process. For example, using a traditional process, such as waterfall, requires a more formal process of requirements specification and management than does an agile process. Since there are various agile processes we will attempt to approach agile requirements specification from a scrum standpoint except when it is noteworthy to make a distinction.

Requirements determine what the product needs to do to solve a customer problem. Some of the types of requirements are scenario, quality of service, safety, security, functional, operational, and interface. A requirement would begin in a Proposed state, then when accepted, it moves to the Active state from which tasks are created, which once the tasks are completed and tested, it moves to the Resolved state, and then finally to the Closed state after validation. This is how work item state transition is defined for the MSF work item template for a requirement. You can customize your work item state to match your process and workflow.

Depending on your process you will create a work item for requirements as a scenario (functional requirement for MSF Agile), quality of service (non-functional requirement for MSF Agile), as a Story (XP), use case (RUP), or as a requirement (MSF for CMMI or Traditional). Work items are used to capture and track requirements. This enables them to show up in your backlog, be ranked, audited, assigned and reported.

Regardless of the methodology used, however, requirements go through an evolution from the business level to the system level and then to the implementation level. In a Waterfall model, each level is completely specified before evolving to the next level. In an iterative or agile model, each individual requirement can go through its evolution independent of the other requirements. This guidance will demonstrate how to specify the requirements for each level with additional mention of supplemental information or detail that needs to be specified in addition to the work items.

> **Note:** The guidance for requirements specification found in this section presumes that a business level requirement, implemented with a “Feature” work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a “Feature” work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Specifying Requirements (the Basics)

Regardless of which point in the hierarchy that one is eliciting requirements (Business, System, or Implementation), there are some core features of Team Foundation Server 2010 that we need to understand mechanically for specifying a requirement and its validation.

Creating Work Item for Requirements

To create work items you can use Microsoft Project, Microsoft Excel, Team Foundation Client, Web Access, or your own tool using the Team Foundation object model. The steps for creating a work item using the Team Foundation Client and entering your requirement is as follows:

1. Select your project in the Team Foundation Client.
2. From the menu bar select Team | Add Work Item
3. Select the type of work item – User Story, Quality of Service, Requirement, etc…

Depending on your process this task would be executed by the role of Product Owner or Product Management (i.e. Business Analyst, Product Manager, Subject Matter Expert, or Sponsor).

Using MSF for CMMI as an example your requirement work item would resemble the following screen shot in Figure 1.

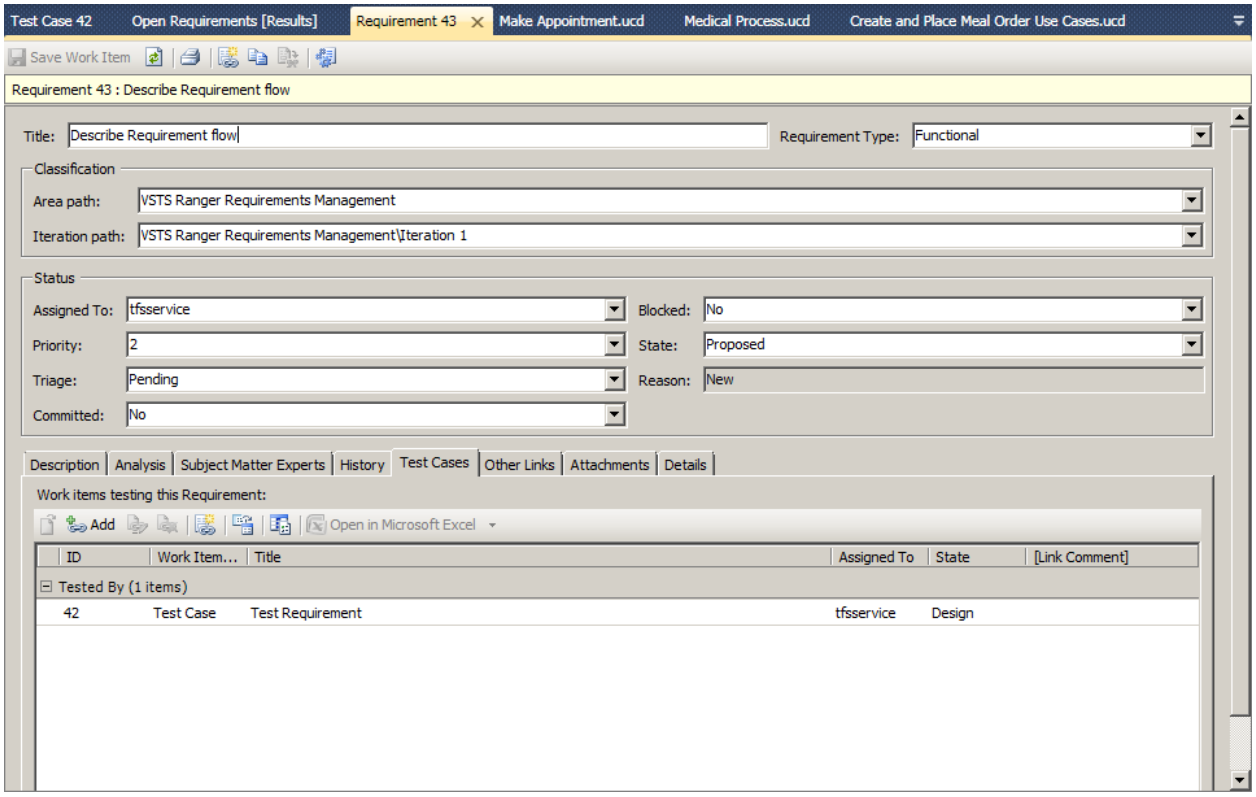


Figure 13: Creating a Requirement Work Item

The Test Case Work Item Type is a new feature in Visual Studio2010. This work item type (WIT) can be linked within a Requirement WIT type as follows:

Linking Test Case to a Work Item

Once you linked your requirements WIT you can then add a link to the test case. To link the test case to a work item: open your requirement work item, select the Test Cases tab, click the Add button.

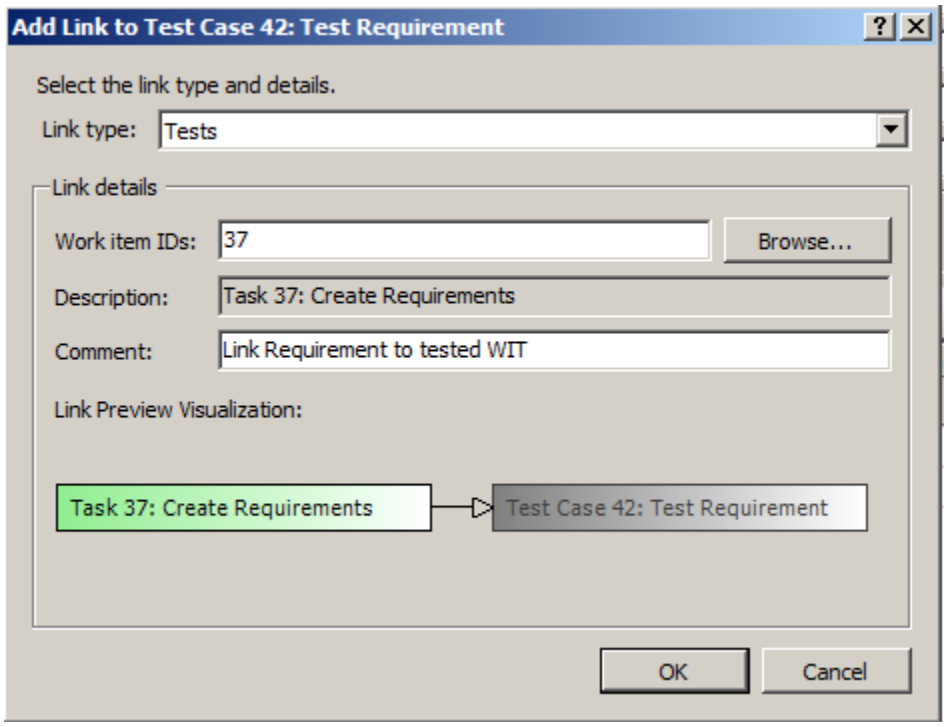


Figure 2: Add a test case to a requirement WIT

Using Figure 3 as a guide, select Tests as Link type and browse to the Work item ID. A comment can be added for transparency. Below that on the dialog box is a visualization of the Task work item and its linked Test Case. After all details have been entered, click on OK. Now your work item should resemble Figure 4.

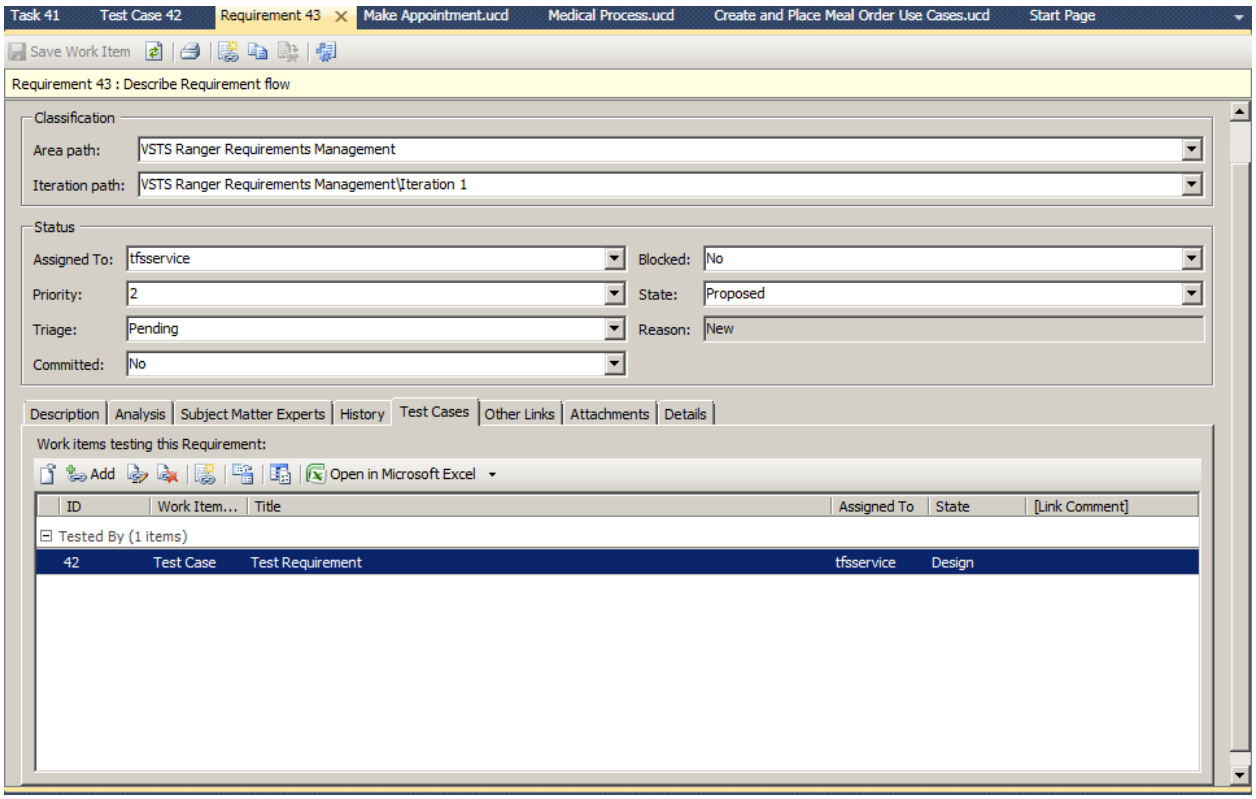


Figure 3: Adding a Test Case to your Requirement work item.

This procedure demonstrated specifying a requirement work item and, then for validation, one of its test case work items at the same time. This same procedure is applicable regardless of work item type or link type. The basics of specifying and tracing work items works for each level through the evolutionary hierarchy.

Scope Specification

In the Requirements Analysis and Breakdown topic area, we described a process for eliciting business requirements as the Feature Work Item Type. This guidance will describe the mechanics for specifying Features and their links to system level requirements, User Acceptance test cases, and additional details found in documentation stored in the SharePoint Portal.

Regardless of methodology, the scope for a project is dictated by the new or enhancement features and their high level details and estimates. The feature is captured as a work item, its details as individual word documents, visio diagrams, PowerPoint presentations and other files. Validation of the feature is captured as a Test Case work item, and system level requirements captured as Requirements (MSF for CMMI) or User Stories (MSF for Agile).

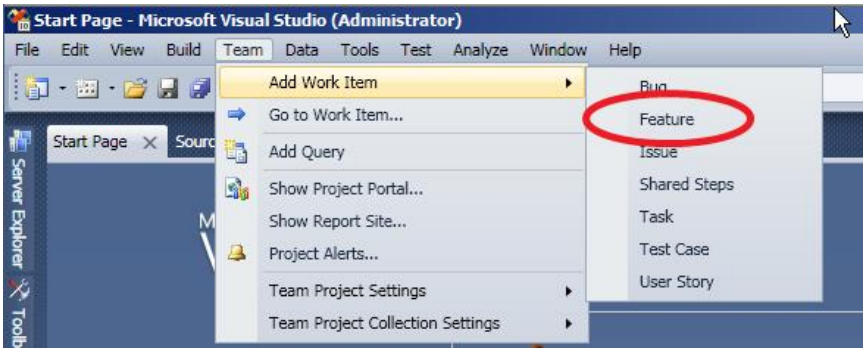
If using the MSF for CMMI template, the Feature work item type was added to the template as part of the Team Foundation Server 2010 release. If using the MSF for Agile template, Features don't exist, so it

is our recommendation that business requirements are added as a Feature work item type in order to establish business scope to system scope traceability.

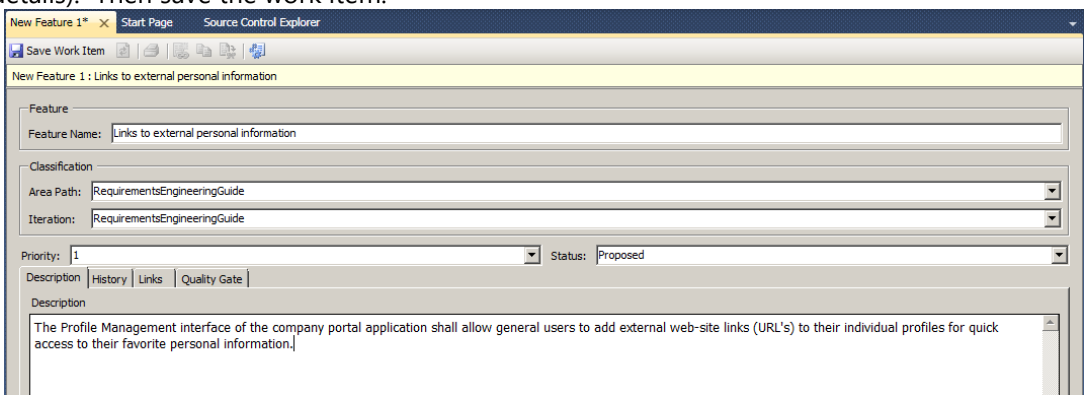
> **NOTE:** Again, adding the 'Feature' work item is not required. It is only a measure in larger organizations that aligns the team, in this case the agile team, more closely with the business stakeholders and IT governance bodies in an organization. An agile team can successfully execute an application development project without the 'Feature' or business level requirements explicitly called out.

Procedure:

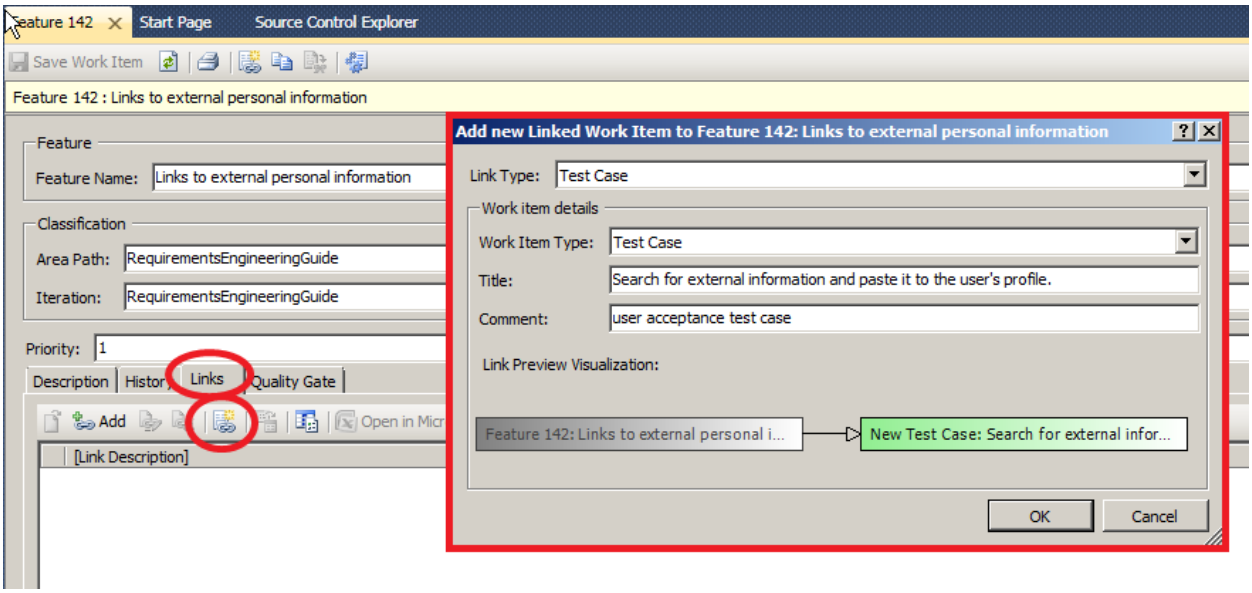
1) From the menu bar select "Team→Add Work Item→Feature



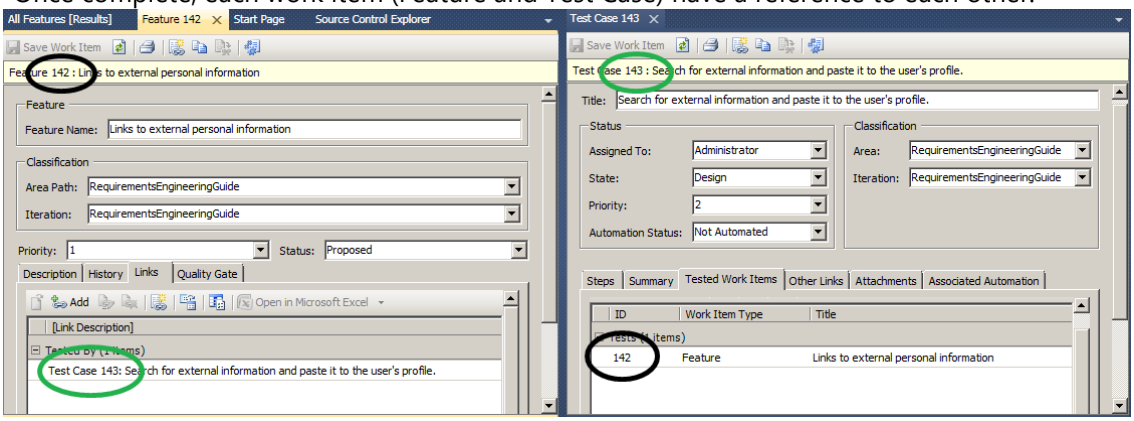
2) Enter all of the field details. To ensure that the requirement is well written, use elicitation checklists that support the scope elicitation effort. (Refer to Elicitation guidance topic for more details). Then save the work item.



3) In order to validate the feature, create at least one user acceptance test by selecting the links tab under the feature detail and adding a new linked work item. In the ensuing dialog, enter "Test Case" as the link type and "Test Case" as the work item type. Then give the test case a relevant title and a comment about the user acceptance test case and save.



4) Once complete, each work item (Feature and Test Case) have a reference to each other:



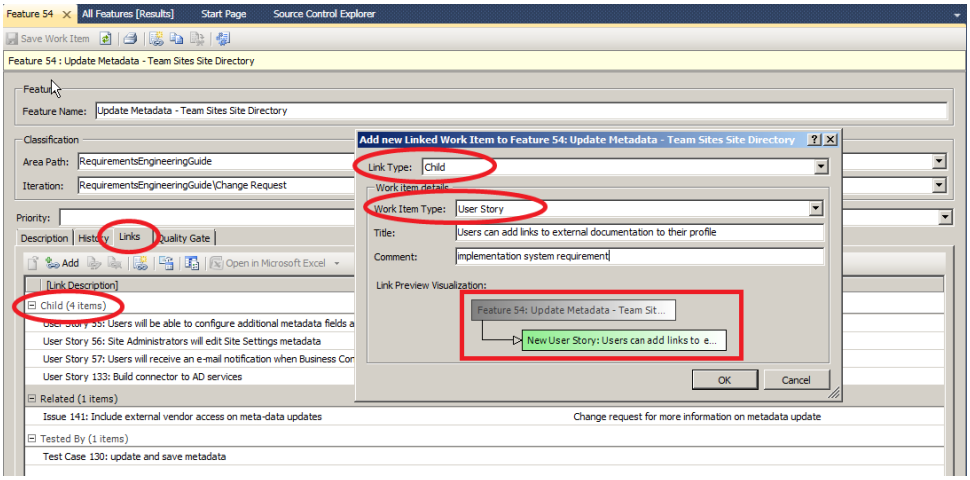
The details of the test case are specified in Microsoft Test and Lab Manager

System Requirements Specification

System requirements represent those requirements for which the team will perform its design and implementation. In MSF for CMMI, these are represented by the Requirement work item which represents both functional and non-functional requirements. Their delineation is determined by selecting a specific value from the Requirement Type attribute of the work item. In MSF for Agile, the User Story work item represents the functional requirement as well as the non-functional requirement. In the previous release of the process template, the "Quality of Service" work item represented non-functional requirements. The reason for the consolidation was to align more closely with the user story metaphor of many Agile Methods. The user story represents "Something that must be implemented".

So, in specifying the system requirements, use the following procedure to identify the work items that represent them.

1) During analysis of each feature, determine each of the system requirements that must be implemented and specify them. Select the feature for which you are performing analysis. Select its "Links" tab, then select the tool icon to "Add new linked work item…". This brings up a dialog to create a linked work item. Select "Child" as its link type and "User Story" (if using MSF for Agile) or "Requirement" (if using MSF for CMMI) as its work item type. Notice the visualization of the link represented at the bottom of the dialog. The implementation link type is represented differently than the Test Case link type shown above. For more information about the new link types represented in Team Foundation Server 2010, refer to the Traceability and Reporting topic area.



Notice also in this view that the system requirements (here, user stories) are shown as "Child" links. The test case that we created for the feature is shown as a "Tested By" link type. This new taxonomy provides a more efficient means for analyzing and performing the work.

2) The next step is to identify the System Test Cases for each of the system requirements identified. This procedure is no different from the one specified above for Scope Specification, so refer to the graphics represented above.

Implementation Specification

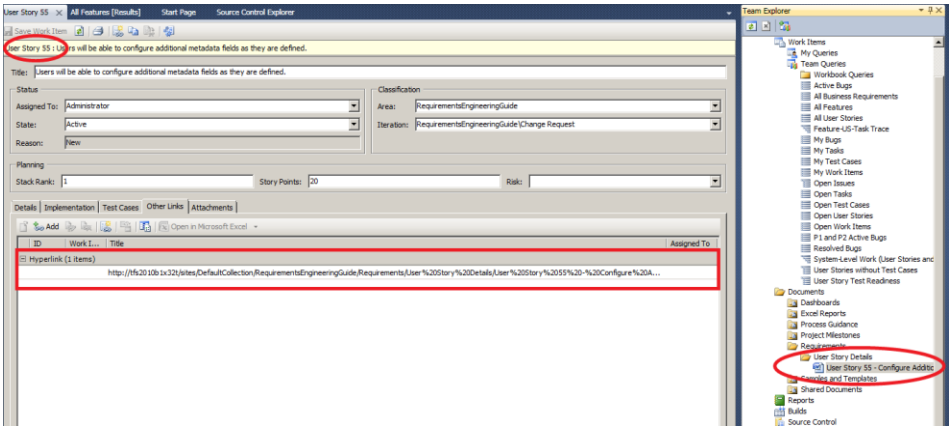
Implementation is performed by the team fulfilling a series of tasks that will either develop source code and unit tests, test cases and scripts, documentation, or other project deliverables that accomplish the goal of finishing the project. The "Implementation" link type is the link type that is used for specifying the results of implantation analysis; analysis performed to identify the implementation tasks and their estimates. We used the implementation link above when we specified the system requirements for the features. It shows up as a "child". Actually, there are two parts to the implementation link type; the child of the work item being implemented and the parent of the requirement that fulfills the implementation. Regardless of nomenclature, the link type allows us to visualize a requirements hierarchy of parent←→child←→grandchild←→ etc.. work items in a tree hierarchy work item query view.

During analysis of the system requirements, specify “Task” work items as the children of all of the system requirements using the procedure described above for System Requirement Specification. Tasks don’t necessarily require tests attached to them, because the work of developing the source code will be linked to the tasks in order to fulfill their traceability. By its very nature, code development should also include unit test harnesses for verifying that the code does what was designed.

Another part of implementation specification is to specify the details of the system requirements. There are a few ways to accomplish this in Team Foundation Server.

1) Describe the details of each system requirement in its “description” field as text. – This is sometimes good enough for clarity, but, often times, our requirements can become quite complex and graphics or lengthy descriptions and scenarios need to be drawn or written to gain that clarity. For this reason, the next bullet describes a more effective mechanism for specifying that detail.
2) Describe the details of the system requirements in a separate file.

   If the detail is embodied in UML, a UML Project in Visual Studio fulfills that objective. Refer to the Analysis and Breakdown topic area for more detailed information for creating UML designs specific to the requirements. Visual Studio architect edition provides mechanisms for creating Use Case diagrams, Activity diagrams, and Sequence diagrams. Expressions Sketchflow provides a mechanism for designing wireframe storyboards and web implementations. Then Word, Excel, and PowerPoint are other file types that can provide detail for a system requirement. One of the more popular uses is a User Story or Use Case template that adds the “flow of events” detail to a word document. Store the document in a document library for the Team Project and then link its SharePoint URL to the requirement using the “hyperlink” link type.



Final Thoughts on Specification

Requirements Specification is very tightly coupled with performing elicitation, analysis and design, and change management. Because of this, some of the specification topic area is covered more deeply in other topic areas of the Requirements Management guidance. Think of specification as the mechanics behind the conceptual process activities of the other process areas. The mechanics we show here are

some of several mechanisms that can be used to specify requirements for your projects using Team Foundation Server 2010.

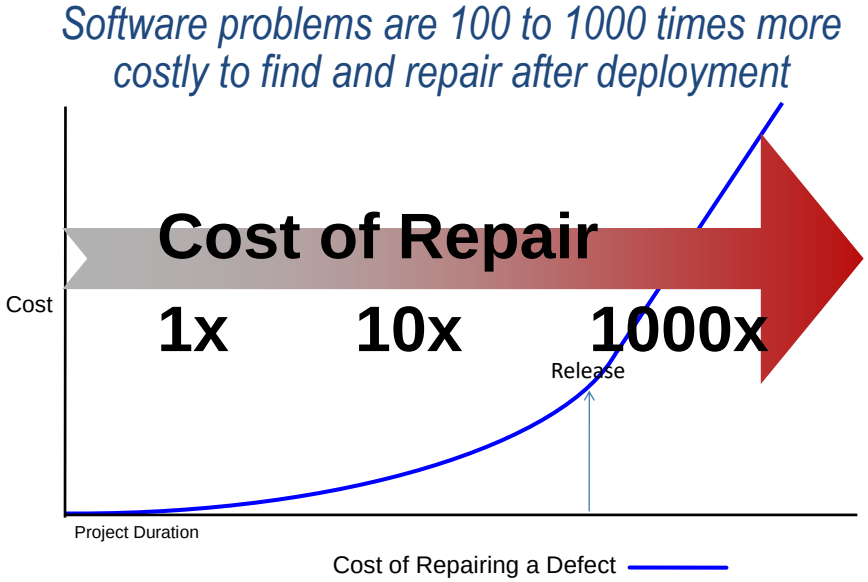
Requirements Validation

"Validation ensures that the requirements exhibit the desirable characteristics of excellent requirement statements (complete, correct, feasible, necessary, prioritized, unambiguous, and verifiable) and of excellent requirements specifications (complete, consistent, modifiable, and traceable)." - Software requirements second edition by Karl E. Wiegers

Validation is a process to assess if the end product is going to satisfy customer requirements. Validation assists in ensuring that requirements are not misunderstood. This approach to delivery has recently been described as "Test-First Development" or "Requirements-Based Testing". Other terminologies have also been described. With validation a practitioner will establish acceptance criteria to gain agreement with their stakeholders on what will be delivered. This includes validating business requirements, functional requirements, non-functional requirements, use cases, analysis models, prototypes, etc.

This, as any other part of the requirements development activity, is not independent of the other activities and needs to be carried out iteratively. The inputs for validation are can be as small as a single requirement or as large as a complete set of requirements described in specifications. The specifications, once validated may result in correcting or closing gaps in elicitation, traceability and analysis.

Validating requirements and correcting those at the beginning of the project help reduce the cost and time spent in correcting these at a later point in the SDLC. This also requires involving the right stakeholders at the beginning of the project who can validate the requirements.



All types of requirements need to be validated including business requirements, functional requirements, and derived requirements such as Qualities of Service or non-functional requirements.

> **Note:** The guidance for requirements validation found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Techniques

Ensure that the resulting product will perform as intended in the user's environment using multiple techniques as appropriate. These are: test plans, checklists, inspections – both informal and formal, and workshops. Workshops provide the facilitated means by which the team can tap into their peers to brainstorm the validity and accuracy of their requirements. Refer to the Requirements Elicitation Topic Area for more specific guidance on workshops as a technique.

Test plans

The V-Model is an industry standard model for establishing test management through all levels of an application development lifecycle. The principles of it can be implemented in a waterfall or any variant of iterative development.

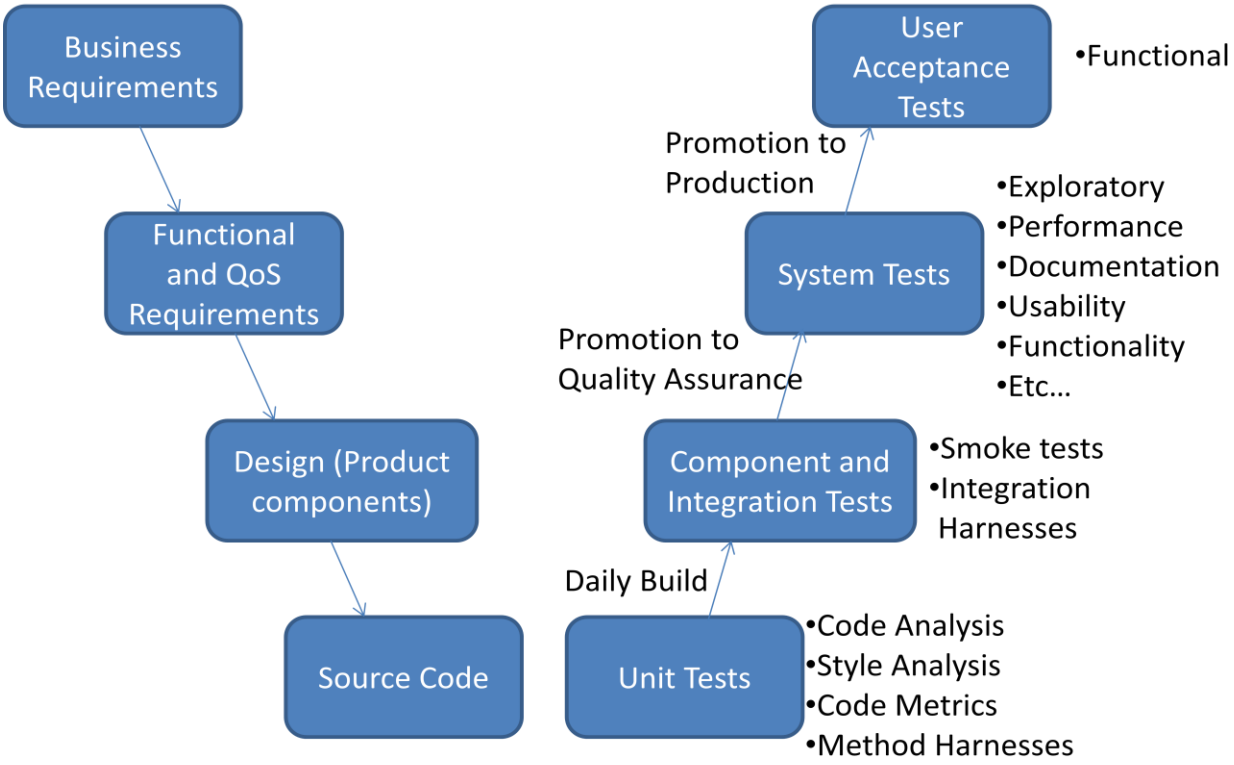


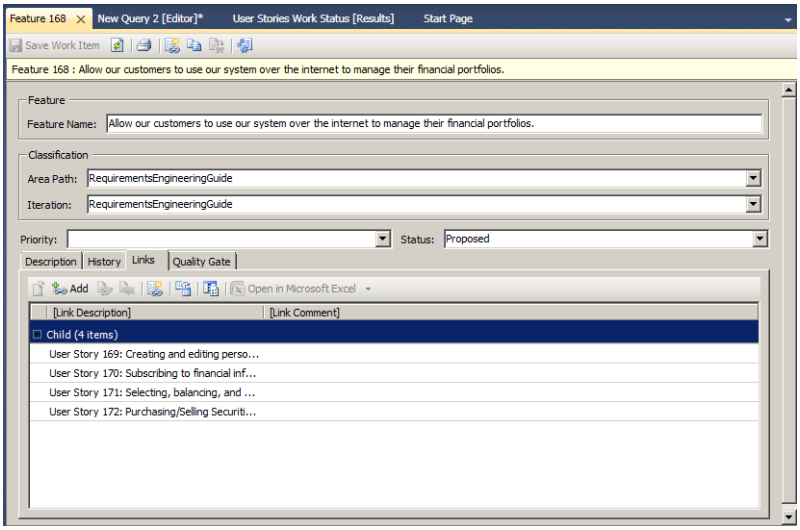
Figure 14: V-Model for Test Management

During the development of each level of requirements on the left side, validation exercises using the techniques described above will assist in developing the tests that will be used for verification after the application code, component, or system is delivered. The best process to follow is "Test First Development" or "Requirements-Based Testing" where an asset is not considered "DONE" until its defined test executes and passes successfully.
So, looking back at the "V-Model" above, the following scenario can be applied to the evolution of all of the elements described:

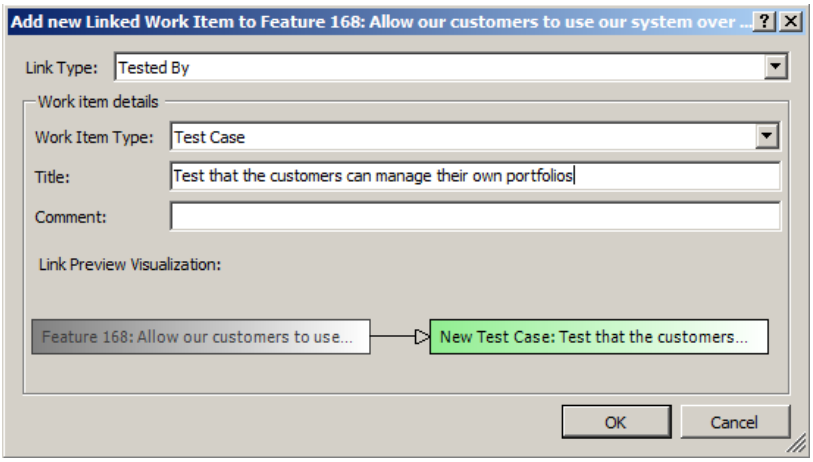
1) A business requirement is defined, but it is not ready for analysis for functional and quality of service requirements until its acceptance tests have been identified and validated.

   Identify User Acceptance Test Cases and link to the Feature Requirement

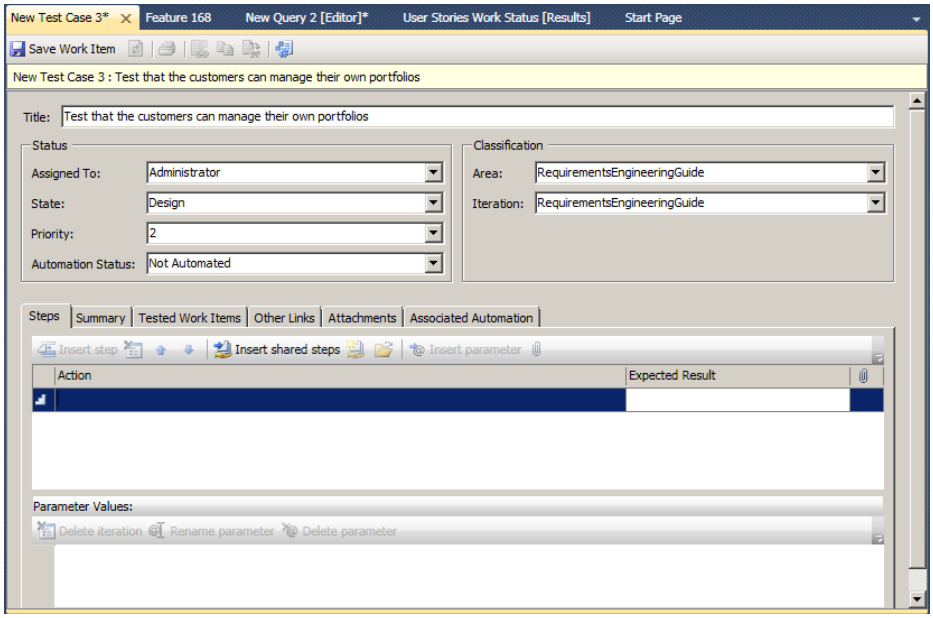
   a) Open a Feature in the Team Foundation Server Team Project to open its display form. Select the "Links" tab to see the children user stories (if you open a feature from MSF for Agile process template).



   b) From the Team menu item or tool bar on the Links tab, select "Add New Linked Work Item …". In the resulting dialog box, select "Tested by" for the link type and "Test Case" as the work item type. Then add a title for the new work item. It wouldn't hurt to add a comment either.

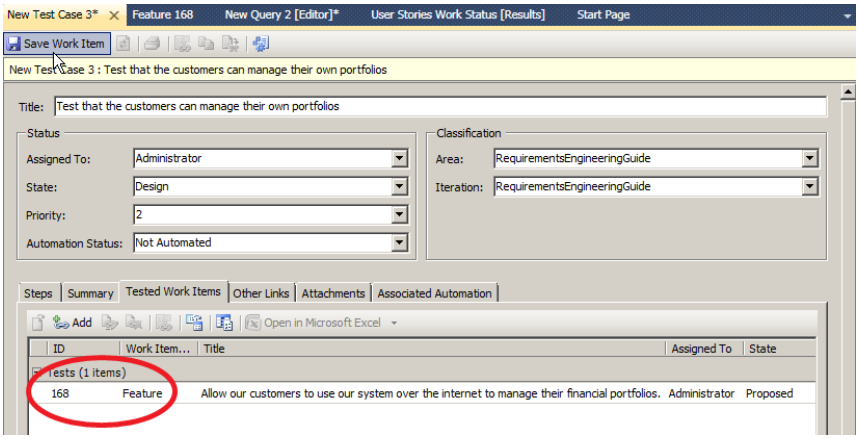


c) The resulting work item has detail that allows the tester to plan, design, and track the development of the test model for the project. The "Steps" tab is where the user will enter procedural steps to execute during a manual test. We will not show it here, but this work item, when opened in the Microsoft Test and Lab Manager client, will allow you to launch your application under test while viewing these steps and mark each step's success or failure as well as the observed results (which may include the call stack!).

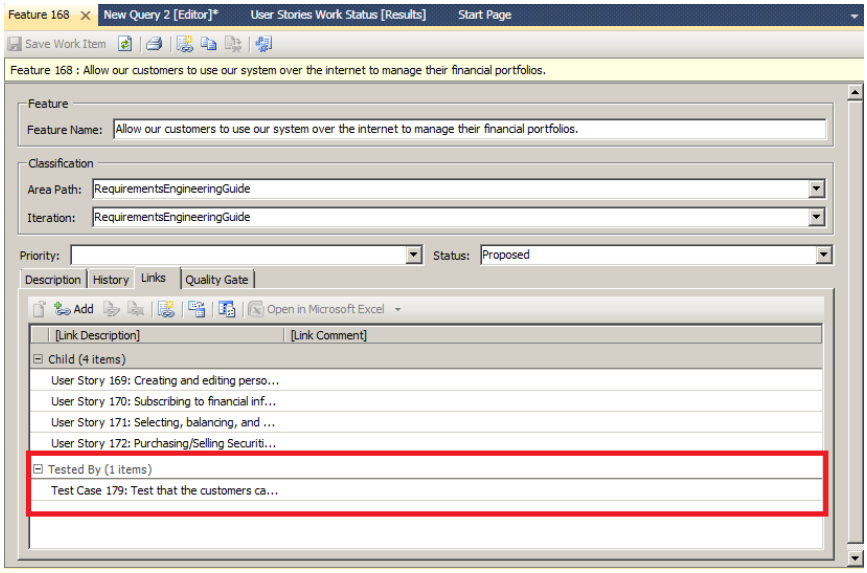


d) The Summary Task is the same as any other work item's description field. This is where you will enter a high level summary of the customer's expectations. A good way of ensuring a clear understanding is to ask the customer: "If I deliver this feature, what will you see that proves I did it correct?" The customer's response, with a little of the analyst's coaching, should be a significant description of the acceptance test procedure.

e) The Tested Work Items tab shows the link to the Feature that we just linked from:



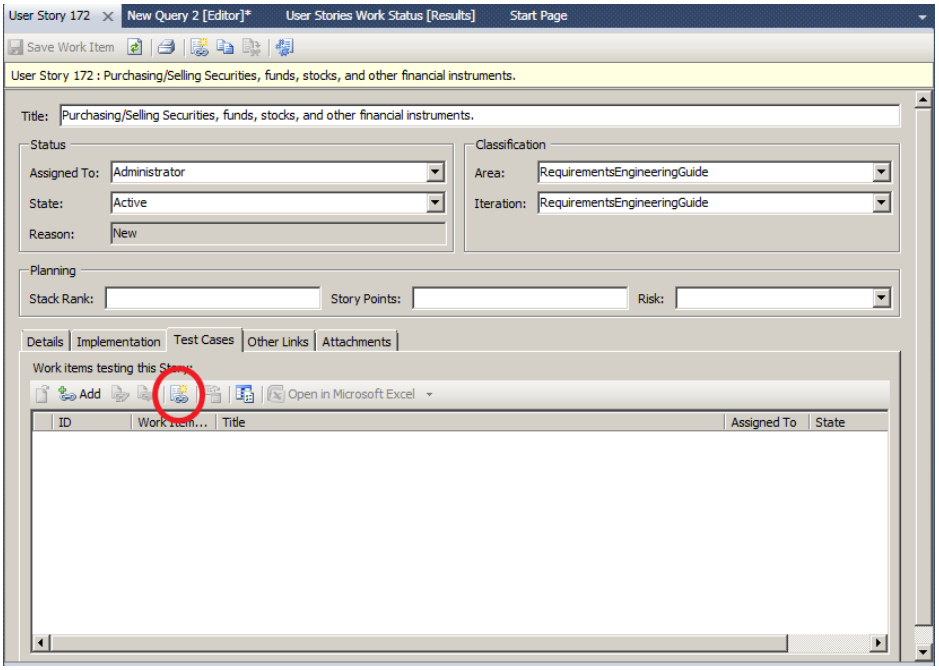
f) Save the work item and then Double click on the feature and select its links tab to view the new test case linked to it.



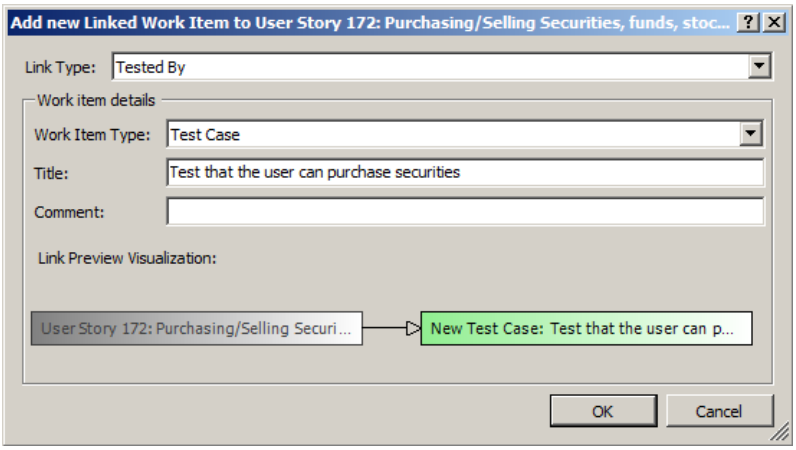
2) The functional and quality of service requirements cannot be turned over to the designers and architects until their test scripts are identified (not necessarily written) and validated.

Describe the High Level Steps of the Test Case

a) Open a child requirement of one of the feature work items. It will be either a "Requirement" or "User Story" work item, depending on the process template being used. Select the "Test Cases" tab to begin identifying test cases. Select the "Add New Linked Work Item …" button as shown in the view below.



b) From the resulting dialog box, enter a title for the new test case. The only options available for Link Type are "Tested By" and Work Item Type is "Test Case", so you don't need to do anything with those attributes.

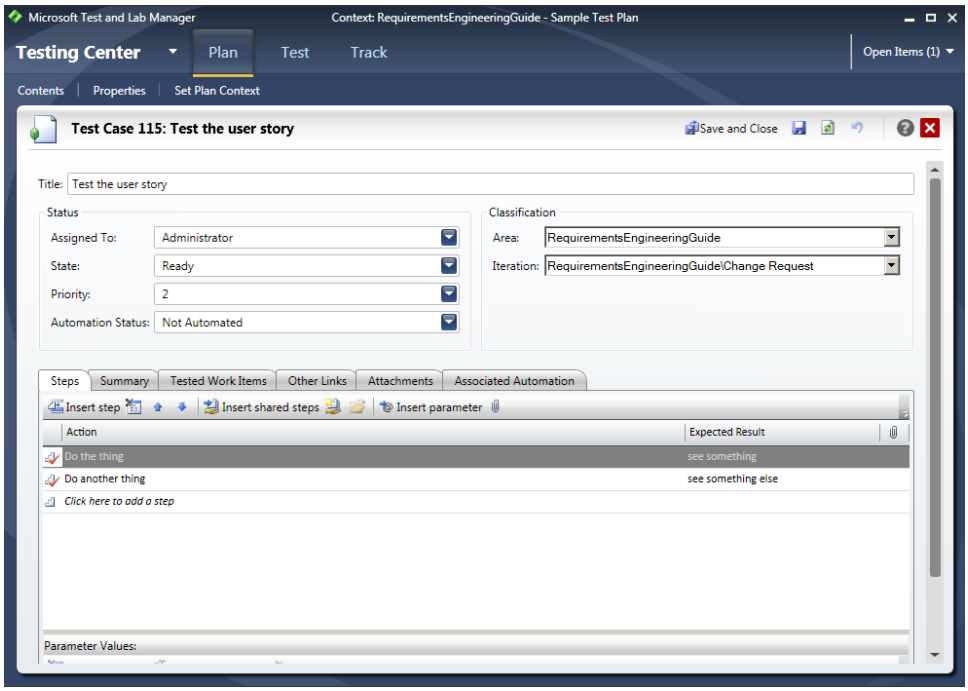


c) The resulting work item is the same as the previous procedure and is linked to the user story in the same way as the test case linked to the feature above. The only difference is a conceptual one. The test case linked to the feature is a User Acceptance Test case and the one linked to the system requirement work item is a System Test case.

3) The application design cannot be turned over to developers until each component has integration and "black box" tests identified and validated.
    a. The developer will write unit test harnesses for their code stubs with the goal of writing code that passes the tests
4) The code then is written along with unit test harnesses that must pass before we can travel up the right side of the model.

5) Once the unit tests pass, the component that encapsulates the tested code can then be tested. We can do this efficiently because the component integration and black box tests were defined during the design activity.
6) Once the component integration and black box tests pass, the system functional and non-functional tests can be executed. Again, we can do this efficiently because the functional and non-functional tests were identified and validated as part of the requirements specification activity.

    Because we've described functional steps in each of the test case work items, we can then view them and build test suites in the Visual Studio Test and Lab Manager client.



7) Then, once the system tests pass, the user acceptance tests don't need to wait, because they will have already been defined and validated during the business requirements specification activity.
    a. User acceptance test cases can also be packaged into a test suite in Visual Studio Test and Lab Manager.

Though the steps of this procedure may imply a waterfall model for execution, really need not be. Each step needs to be executed for an individual requirement, component, or line of code at that level. In this way, an agile metaphor strongly applies to this model.

UAT test cases are to be created based on the business and functional requirements defined and signed off. These are to be developed or at least reviewed by end users who have provided the business requirements (or features) to begin with and focus primarily on the functionality.

As you develop these test cases, you primarily validate the requirements to be verifiable. When you establish that the test cases are complete and correct, you establish that the requirements are

developed and documented to satisfy the customer needs. When the test cases pass, you establish that the system is built to satisfy customer needs. One should make sure that the test cases tie back to the functional requirements and no requirements are missed out.

It is the business users who should provide the prioritization and ensure that the requirements are verifiable at the business requirements level providing clarity as the requirements are further developed.

It is expected that the requirements will have ambiguities at this level in terms of how they will be detailed, which are resolved as the requirements are broken down and analyzed.

In many traditional development methodologies, practitioners are trained to remove all ambiguities before analyzing to the next level. For example an ambiguity identified for a business requirement must be removed before the analyst will derive its functional or non-functional requirements.

In an agile methodology, the ambiguities are acceptable so long as there is enough agreement to continue working and test something that can then be reviewed. The agile methodologies will take a more loosely defined approach to validation because the methodology accepts that requirements will evolve, change, or even become obsolete during the project. This is not to mean that the validation exercise, itself, is less rigorous. In fact, agile teams execute validation more rigorously due to the fact that they are constantly reviewing progress and gaining agreement on the outcomes.

Once the Functional and QoS requirements are developed, system tests cases are built. These cover the functionality in detail including the usability aspects and conformance to the usability requirements, and non functional requirements including availability, auditing, operations and management, data management, performance, recoverability, scalability, security, and usability. Other non functional requirements like testing, portability, localization, documentation can also be covered. Further sections describe how the various non functional requirements can be validated if these are stated such that they are verifiable.

The model drawn below represents the Team Foundation Server structure that supports the V-Model described in the previous drawing. As you can see, a “Feature” (or a business requirement) as a counterpart test defined in a Test project in Visual Studio. Team Foundation Server allows the user to establish a link between the work item and the test within the test project. Even tighter integration is established when the test engineer checks the test project into version control. Upon check-in, associating work item to the change set can be forced using Team Foundation Server Check-In Policies.

A similar mechanism in Team Foundation Server supports “Scenario” (functional) or QoS (quality of service or non-functional) requirements.

When it comes to development of code, our guidance is to establish tasks for each of the many parts that need to be implemented to support an individual requirement. Using this method, a one-to-one relationship can be established between a small unit of development, and it’s tests as well as their

linkage to a single task. In the Requirements Traceability topic area, we'll describe more detail about the specifics to this linkage and the reports that can be derived from it.

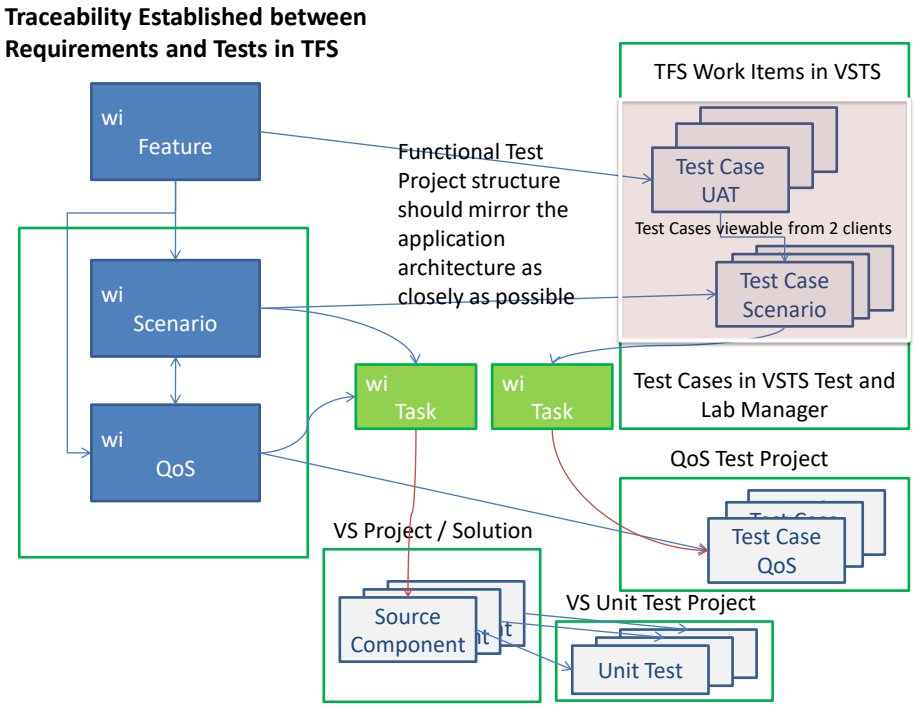


Figure 15: System Test Management Traceability

Once design and source code are developed, corresponding integration test cases and unit test cases are developed. These test cases help more towards verification than validation of requirements.

Test Planning (Component and Unit Level)

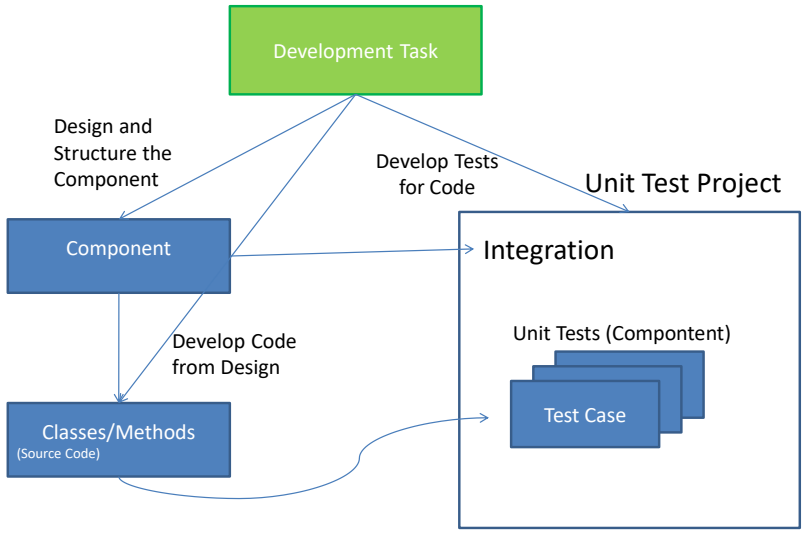


Figure 16: Unit Test Planning and Management

Checklists

One can build various types of checklists to establish if the requirements exhibit the desirable characteristics of well-formed requirement statements (complete, correct, feasible, necessary, prioritized, unambiguous, and verifiable) and of well-formed requirements specifications (complete, consistent, modifiable, traceable, etc…).

Typical checklists used for validating requirements at all levels are as listed below. Samples of many of them accompany this guide in "Requirements Engineering - Checklists.xlsx". Each is represented as a separate worksheet in the workbook.

- **Product Acceptance Checklist** – Used as milestones to ensure that all requirements have been covered by the required documentation, design, code, tests, and other critical needs.
- **Requirements Review Checklist** – Used as a guide while reviewing a business requirements specification, functional requirements specification, or technical requirements specification. It asks for evidence that documentation standards have been fulfilled and that the requirements are well-formed (as described above)
- **Requirements Style Checklist** – Similar to the previous one
- **Use Case Review Checklist** – Used to review a scenario requirement and its detail which should include the steps of all sub-flows, graphical representations of them, and any supplemental data that accompanies them. The same checklist applies in the context of agile stories or traditional functional requirements.

- **High Level Design Review** – Used to ensure that the requirements are implemented, at a high level, against the selected architecture.
- **Acceptance Test Planning, Results, Change Requests, WBS**, and other checklists fulfill other requirements and project coverage responsibilities.

Refer to the accompanying document template, "Non-Functional Requirements Template", as a checklist for identifying and validating quality of service requirements (performance, scalability, business cycle, maintenance, support, disposal, etc…)

Inspection

Both informal and formal inspection can be used to validate requirements. In most cases you will typically end up having informal reviews done during requirements development. For formal reviews, one should build an appropriate team of reviewers from various areas including those building the system and those using the system i.e. considering all stakeholders. The reviewers should review the documents and other artifacts for completeness and correctness while using the checklists identified for the team.

A good process to follow for inspections is the following:

1) Send the material to be reviewed to each of the reviewers independently.
2) Specify what each reviewer is to look for while inspecting the asset. For example, a tester should be determining if the asset is testable. That is, can a test be written and executed to verify its completion. Tell them to highlight the documentation with their notes, comments, and criticisms. If they find things wrong, they should be prepared to identify an alternative to the error.
3) Schedule the meeting giving the attendees enough time to review the material. A week is usually enough time for a work packet.
4) Review only the issues that have been identified. This way, the meeting will efficiently communicate that the reviews were completed without wasting anyone's time. It is important to note that a requirements review or inspection is not a brainstorming and idea generation session. Explain to the attendees the importance of being prepared.
5) Document any discrepancies and define tasks for the author to fix and re-administer an inspection.

Acceptance review workshops should be executed in a similar way that inspections are performed. Essentially, they are the same with the exception that an Acceptance review culminates in some type of signature.

Technology Support

Use Team Foundation Server to support the above activities. Guidance for using Team Foundation Server to store results of inspections and acceptance reviews of requirements. Refer to the Elicitation

topic area for descriptions of document libraries in Team Foundation Server for storage of checklists and resulting work products.

CMMI Specific

"Analyze the requirements to determine the risk that the resulting product will not perform appropriately in its intended-use environment." – CMMI, Guidelines for Process Integration and Product Improvement, Chrissis, Konrad, Shrum.

Requirements Change Management and Approval

"A baseline is a set of specifications or work products that has been formally reviewed and agreed on, that thereafter serves as the basis for further development, and that can be changed only through change control procedures. A baseline represents the assignment of an identifier to a configuration item and its associated entities." – CMMI, Guidelines for Process Integration and Product Improvement, Chrissis, Konrad, Shrum.

Requirements Change Management is performed to:

- Obtain authorization before creating or releasing baselines of requirements and their associated configuration items
- Create or release baselines only from requirements and their associated configuration items in a central repository
- Document the set of requirements and their associated configuration items that are contained in a baseline
- Make the current set of baselines readily available (read-only if not current)

Requirements Change Management and Approval describes how a practitioner will formally capture the approvals for new requirements or changes/enhancements to existing requirements. Focus in this area will be on workflow describing a requirements change process, management of requirements baselines and authorizations to continue on requirements realization. Attention to rigor in this topic area will lead to compliance to industry measures such as CMMI and ITIL and compliance with industry regulations like FDA CFR-21, Part 11 and Sarbanes-Oxley can be achieved.

Within the change management and approval topic area, there are subtle differences between an agile development model and a traditional waterfall model. Despite being designed for managing change, the agile change management and control policies still need to be built in a similar fashion to the traditional approach when it pertains to baselines and approval cycles. The most significant difference lies within "In-Flight" changes. A waterfall model does not provide a mechanism for waiting on a release for changes, where the agile model can. This guide will make note, where applicable, to the difference to the two models. Separate sections are not called out.

> **Note:** The guidance for change management and approval found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Change Management and Approval - Generic Scenarios

In all software development projects, whether being performed iteratively and agile or traditionally, with a waterfall approach, requirements changes should be orchestrated with control and responsibility. What follows are scenarios that need to be addressed for all software development projects.

New Requirements

Typically, Greenfield development is categorized as a new project effort to develop an application from scratch. That is, enhancements to existing systems and applications will not be necessary. In this case, new requirements will just be specified as described in the elicitation and specification areas, ensuring

that proper validation is performed. This scenario requires no further attention to change management. The following scenarios all affect this scenario.

Enhancement Requests

Once a project is started or complete, requests to change the application will be sent to the team. For the purpose of this section, we will only describe control of enhancement requests for the later situation; the project is complete. By this, enhancements are identified to change a system that is in use by production users. A mechanism to control these changes requires the following:

- Change Request items – These change requests should include data required for performing analysis of the impact of the change as well as providing estimates and approvals. They will be treated as requirements that require elicitation, analysis, validation, and specification and are described in those topic areas of this guidance.
- Work Flow – Each requirement will evolve according to the rules and policies established by the organization's methodology and standards.
- Approvals – Stakeholders will approve the initial set of requirements for the initial business requirements milestone of a traditional project, followed by subsequent approvals for each of the functional and technical requirements milestones. Agile projects require a concensus at the beginning of each iteration and approval during the iteration only takes place at the end of the iteration during the application review or retrospective.

Requirements Defects

Requirements defects can be described as discrepancies that arise after the initial set of requirements have been accepted by the team. In a sense, these types of changes are very similar to enhancement requests. A mechanism to control these changes requires the following:

- Change Request items – These change requests should include data required for performing analysis of the impact of the change as well as providing estimates and approvals. These items are related to misunderstandings identified by a stakeholder to the project. Typically, they are changes to an initial requirement, but could also be either the introduction of a new requirement or the discovery that a requirement is obsolete, so it needs to be deleted.
- Work Flow – Each change request should provide a mechanism to investigate, approve, assign, analyze, develop, and verify the new request through the lifecycle. This process takes on more rigor for a traditional waterfall project and less formality for an agile project, even though there are rules for injecting new requirements on an agile team.
- Approvals – In phase approvals are necessary for each state, or 'gate', of the workflow during a traditional project. End Phase, or end of a release cycle approvals are required for control of all projects. Release for a traditional project means at deployment to production at the end of a project, while release for an agile project means that it will occur frequently at iteration milestones throughout the project.

In-Flight Discoveries

Very similar to requirements defects, in-flight discoveries are just that, changes that arise due to analysis, design, development, or testing of a requirement that provides the team more information that may not have been available when the original requirement was accepted by the team. Traditional teams handle in flight discoveries differently from agile teams and we provide those differences below in each respective section.

Change Management and Approval – Team Foundation Server Support for Scenarios

Changes to requirements on an agile project, from a strict "Agile Manifesto" sense, are very simple to control because they are identified and held until the beginning of a new iteration, not dissimilar to identifying enhancements to a project that has completed.

Changes to requirements on a traditional waterfall model, are simple, as well, but only from the mechanics of establishing, comparing, and approving a baseline. The management of changes to in flight requirements is a little more difficult, because a waterfall methodology requires not only the change to a requirement to be evaluated, but that change in the context of ALL of the other project related assets; Project Plan and Work Breakdown, Project dependencies, Qualities of Service, etc... As such, change management and approval to a traditional project requires additional workflow and rigid adherence to its gates. Even with such a policy, predictability is much more difficult to establish on a traditional project.

Preparing for Baseline Management

Requirements are captured using specific work item types in Team Foundation Server (e.g. as scenario or user story work item type). Documentation (specifications and wireframes) are stored in the Team Project site on SharePoint. A hyperlink is created in the work item to the documents on the Team site. These hyperlinks can also be placed on the work item which assigns a specific piece of development work to a developer.

The document library on the team Site is structured to reflect the agile approach that is being followed. In the Requirements library a folder is created for the system of record. It is a good practice to organize the asset types in their own folders. For example, scenario requirements (Stories) should be in one folder and Non-Functional (Quality of Service) requirements should be in another. Documents common to the entire project, such as the product vision, and product constraints or additional non-functional requirements specifications should be stored at the root of the requirements folder.

In order to establish traceability, the story and QoS work items should be linked to the appropriate documents in the Stories and QoS folders.

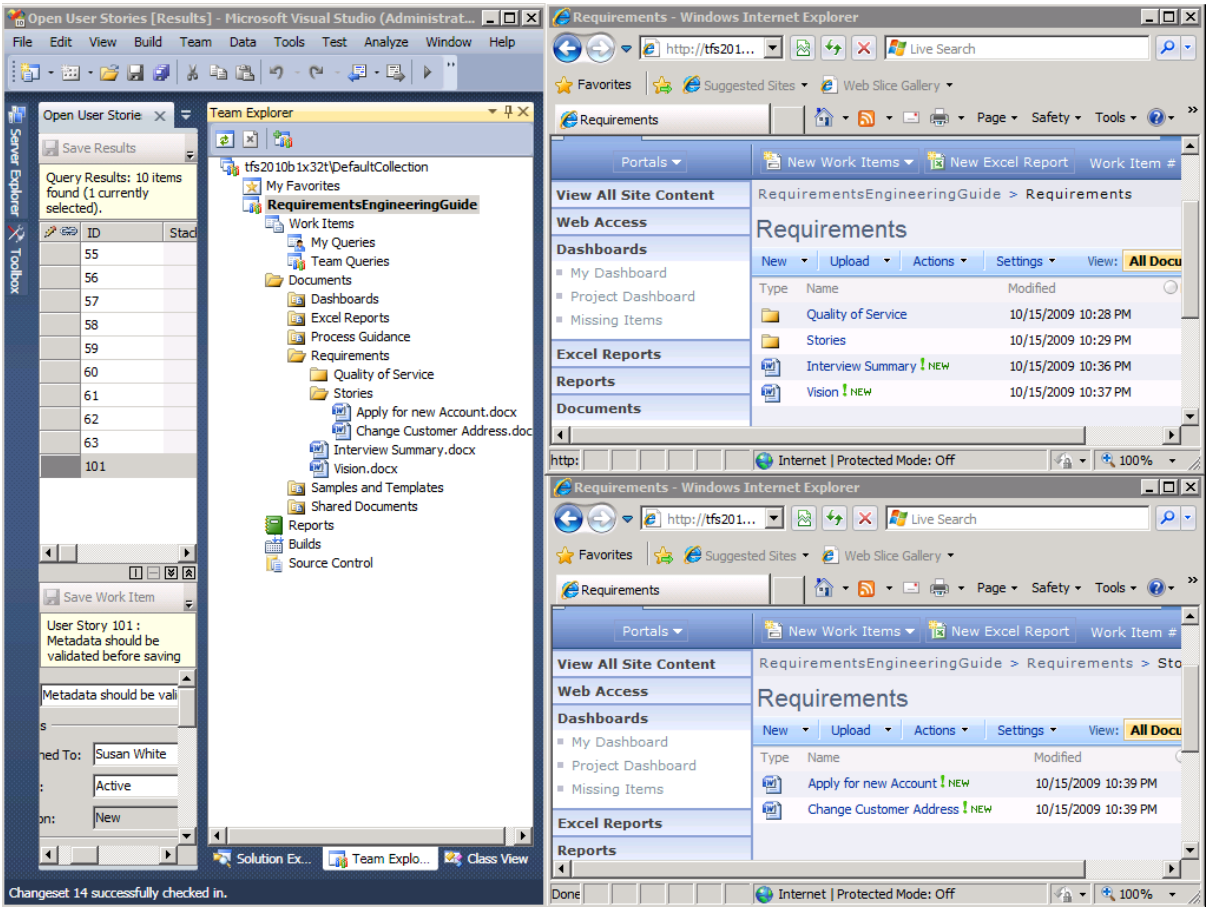


Figure 17: Document Representation in Team Explorer and WSS

In the screenshot above, the Team Explorer view on the left is open to the document libraries showing requirements and stories. The same documents are shown on the right in the two views of the Windows SharePoint Services (WSS) Portal. This comparison demonstrates that Team Foundation Server team members can view work in either application, depending on where they do most of their work. Developers and analysts, for example, might spend most of their time in Team Explorer, while managers and business analysts might spend more of their time in the Portal.

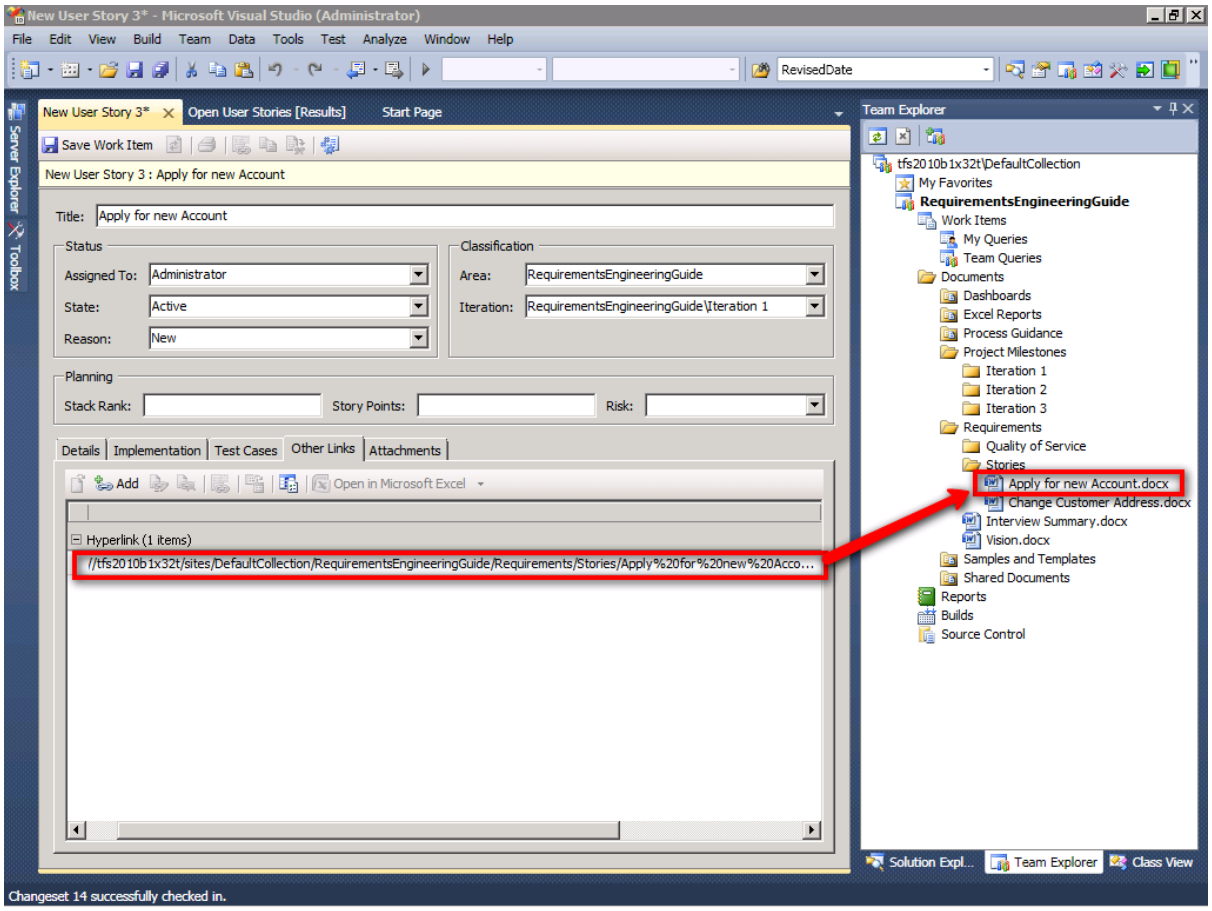


Figure 18: Establishing the traceability link between a work item and a document

The documentation stored on the Team Site is also directly accessible from the Visual Studio IDE. Team Foundation Server provides a robust mechanism for linking documents and other file-based artefacts to a work item. By placing the documents in the documentation library and turning on "Versioning" in the portal, changes to documents linked to a work item via a "hyperlink" link type, will always be reflected as the latest revision on the link.

Team members can compare the latest document to its predecessors using the Word compare functionality which is enabled through the "Revision Tracking" capability or the SharePoint versioning feature.

Structuring the document library for iteration milestones will provide a mechanism for establishing baselines that can be accepted and approved by stakeholders. If you have an ongoing development, such as product development, with different major versions of the product, consider using a root folder for each major version inside the requirements document library.

Notice in the picture above that the requirements are stored by iteration. In this example, Use Cases are being stored, but User Story details or any scenario-type requirement detail can be stored in the

same way. These documents should be linked to the work items that provide their management. That linkage is enabled by specifying the URL for the document as established in the SharePoint Portal.

Striking a Baseline

For any baseline of a set of requirements, the only set of requirements that is important is the one tied to a release. Especially in cases where governance or compliance is concerned, like the Food and Drug Administration, the changes that are made to requirements throughout the development cycle (requirements inception to release delivery) only the settling state of the set of requirements is all that matters. From a baseline standpoint, the final set tied to a build delivered to a release is the source of record for any enhancement requests or defects identified in production.

Now, it is very important to make sure that a customer's or stakeholder's changes are, ultimately, validated, tested, and accepted. Only then are the baselines required. This type of baseline will be referred to as a ***"System of Record Baseline"*** whereas a baseline taken during development prior to the release of a build will be referred to as an ***"In-Phase Baseline"***.

This guidance will describe both types in the following two sub-sections.

In-Phase Baseline Management

Within any development process, whether agile or traditional waterfall, requirements are thrashed out as the project progresses. Just before development begins, the requirements (features, scenarios and wireframes, etc...) are finalised and approved. These approved requirements are then "base lined". Any changes to these requirements now have to go through a change control process. Without the baseline, control is easily lost and changes might be applied haphazardly and without approval leading to scope creep and ultimately threatening the success of the project.

For an agile project, control of changes to these requirements is made by keeping development to short deliverable segments of work. Using 2 to 4 weeks to deliver a small set of requirements, it is easy to allow a team of developers to focus on delivering that set of requirements without any disruptions or changes. Changes that come in while an iteration is being delivered will wait on the product backlog and go through approvals only for the subsequent iteration. Rarely is there a need to stop an iteration due to a change to a requirement. When the iteration ends (whether it completes its schedule or is terminated abruptly), changes to the requirements are approved and the next iteration is planned.

Using the structure that was established in the section above, "Preparing for Baseline Management", a baseline should be established at the end of a significant milestone. For an Agile project, that would be at the end of each iteration.

In order to establish the baseline, extracts of all important artefacts need to be taken and stored in a location established for the baseline. Our recommendation is to define a "Project Milestones" document library with a separate directory folder for each milestone. Here, "Iteration 0", "Iteration 1", "Iteration 2", etc... The important artefacts to consider are the ones that demonstrate the complete trace hierarchy from Business requirements to code and test results established for that milestone:

- Business Requirements Extract – Using a work item query filtered on "Feature" work items with all the attributes that contributed to the decisions for developing the application, extract the entire set to an excel spreadsheet and store in the milestone directory.
- Scenarios Extract – Using a work item query filtered on "Scenario" work items only associated with the milestone ("Iteration Path = Iteration 0" for example) with all of their attributes and extract the entire set to an excel spreadsheet and store in the milestone directory.
- Tasks Extract – Same as scenarios, but only filter on Task work items for the iteration.
- Stories – Each document that represents the detail of the scenarios for the iteration should be copied to the milestone directory. Even though revision histories in SharePoint allow for comparisons of documents from version to version, this copy will make it easier to compare the "baselined" version against the current changes. Without the copy, it could be difficult to know which document is being compared.
- Other documentation – Any other documents, such as design specifications, vision documents, architecture or infrastructure specifications, etc..., should be copied to the milestone directory.
- Build Report – A snapshot of the final build report will establish the linkage between the documentation (work items and files) for the baseline and the source files, tests, and test results for the same baseline. The Build Report is a report that comes standard with both the "MSF for Agile" and "MSF for CMMI" process templates. It is a listing of all of the builds stored in the system. The only one that matters is the final build for the iteration that represents the baseline. This detail should be extracted as a file and stored in the iteration milestone folder.

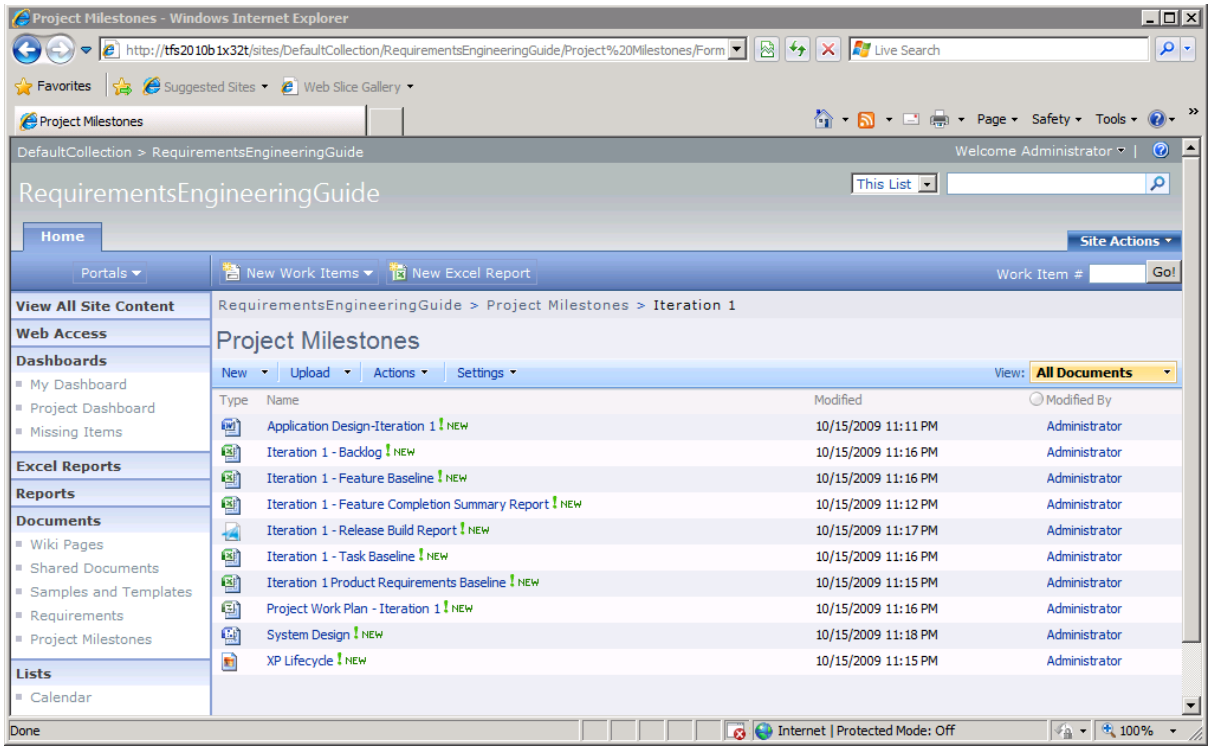


Figure 19: Files stored for an iteration baseline

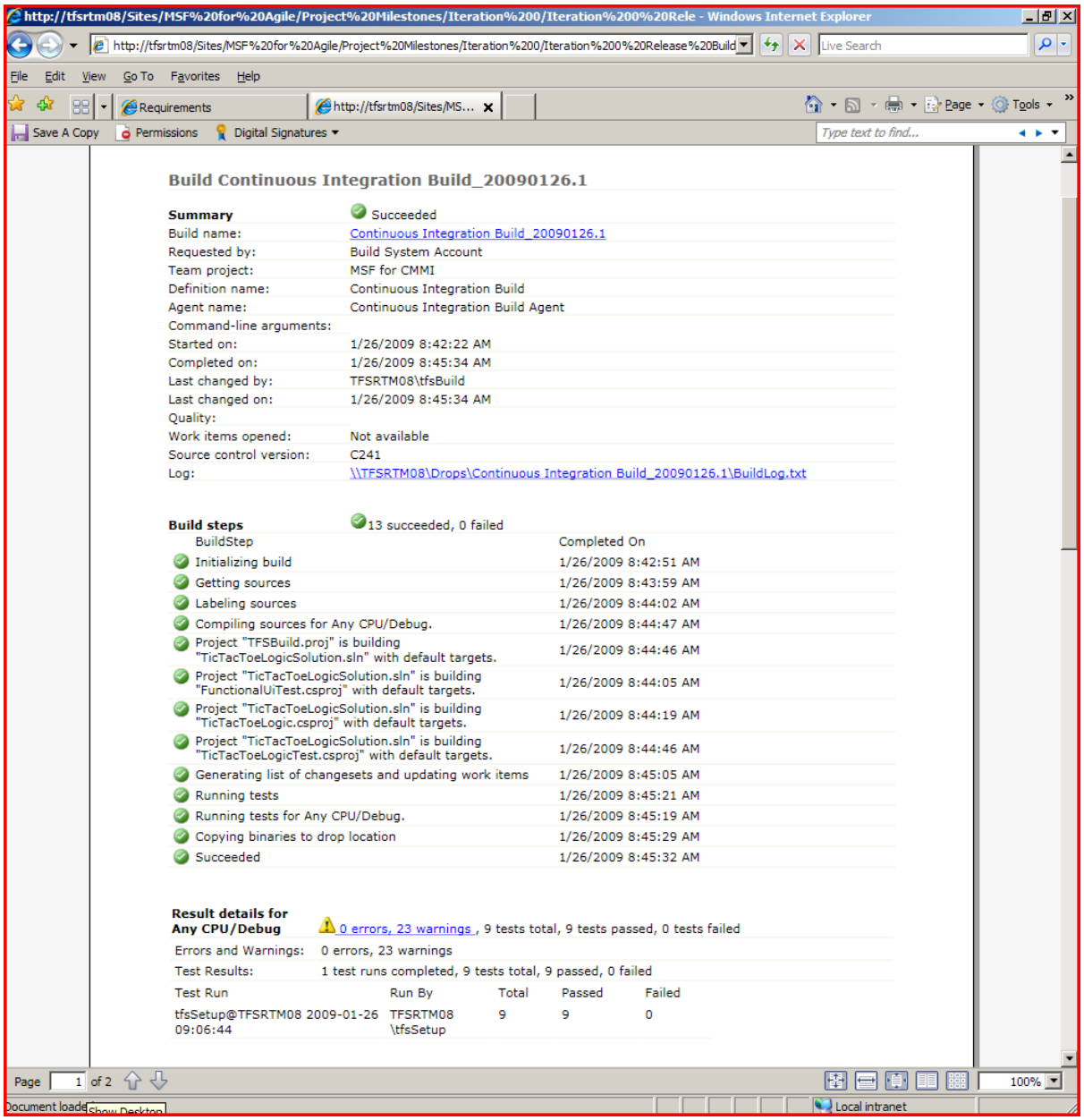


Figure 20: Iteration Milestone Final Build Report

In the build report above, the build is listed, the steps for the build reference the compile and link steps taken, the results of the tests run against the build, including their pass/fail status, and (even though they aren't shown in the picture) the work items and change sets that went into this build (somewhat of a bill of materials).

Approving the Baseline

Many methodologies, especially those rigidly governed for compliance to industry standards (CMMI) or government regulations (FDA, Sarbanes-Oxley), will require review and approval of the baseline. By establishing the baseline as described above, WSS can support this approval cycle.

Approvals can be turned on in WSS for the baseline folder. The administrator should identify a change control board as a user group whose purpose is to identify the authorized approvers of each baseline. This practice, as opposed to identifying individual users, provides flexibility to add and remove members of the board as the organization evolves.

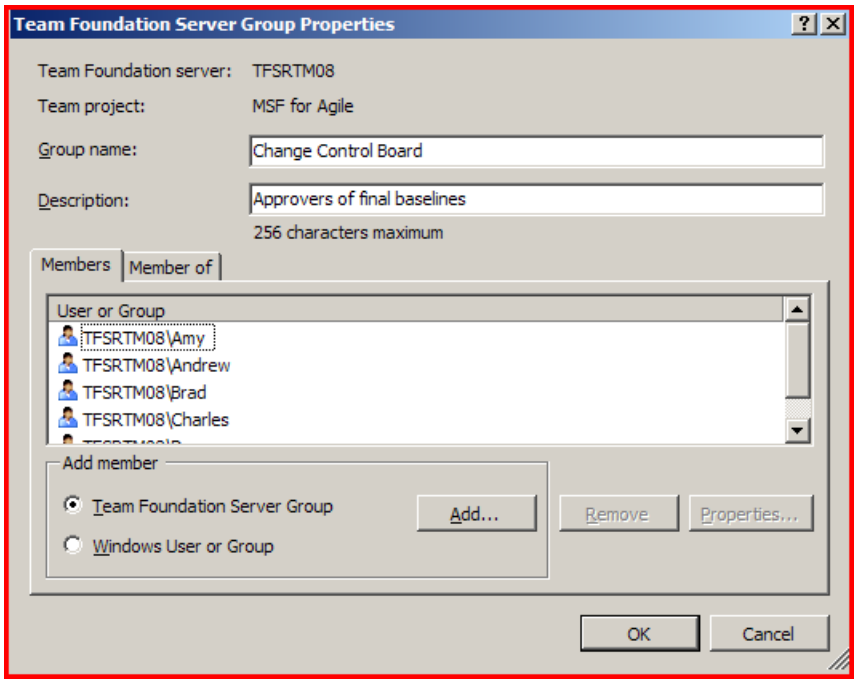


Figure 21: Change Control Board Security Group

With Approval workflow turned on in WSS and a Change Control Board security group allocated to the baseline, approvals can be controlled on baselines, thus establishing the ability to comply with regulatory audits for change management.

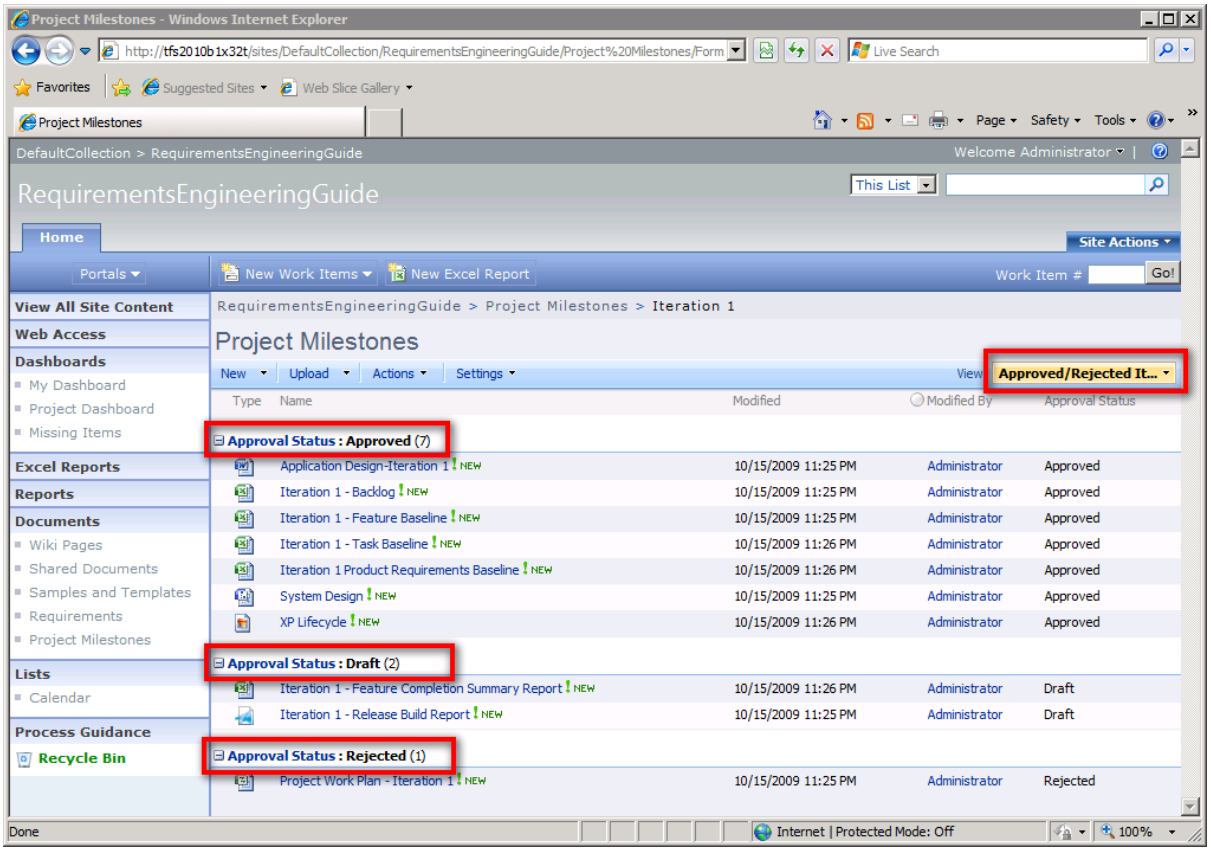


Figure 22: Approve the Baseline

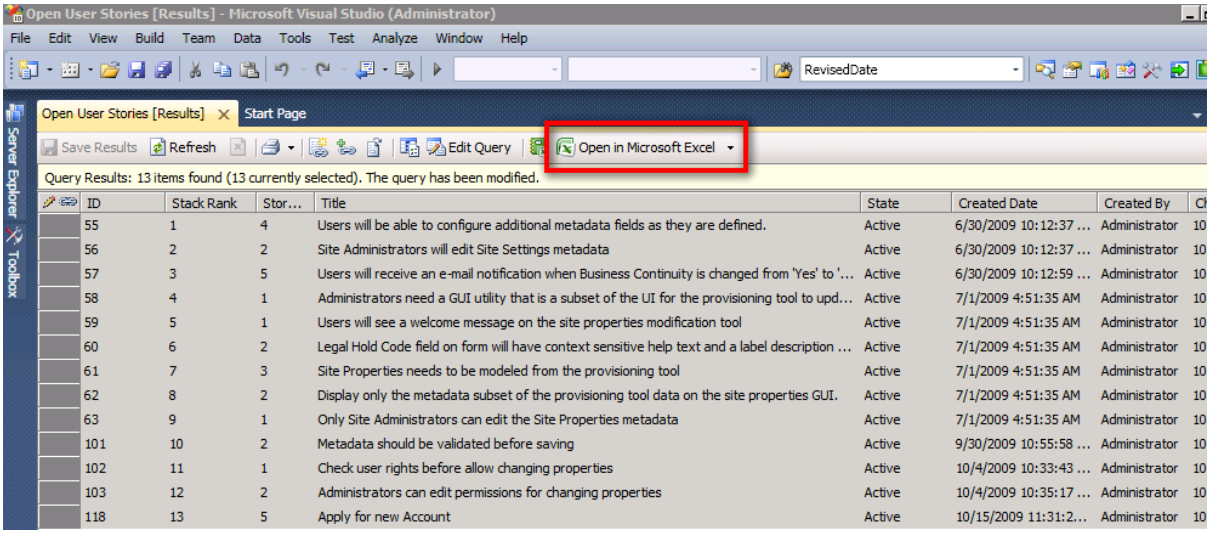
Mechanics of Comparing 2 Baselines

For comparing two baselines, a core assumption needs to be made; a significant milestone has been reached and approved. This can either be the completion of an agile iteration, a gate reached within a waterfall process, or the product has been successfully released. Either way, there is a set of work items that reflect that milestone. Because of the disparities of requirements types and traceability hierarchies used by many different methodologies, we'll use the department of defence methods for our example. During a DoD project process, the following milestones and their associated work items are applicable:

- Business Requirements Document (BRD) Milestone – The business requirements are approved and based.
- Functional Requirements Document (FRD) Milestone – The functional requirements are approved and based.
- Technical Requirements Document (TRD) Milestone – The technical requirements are approved and based.

During an Agile project, the product backlog and iteration backlogs are approved and based.

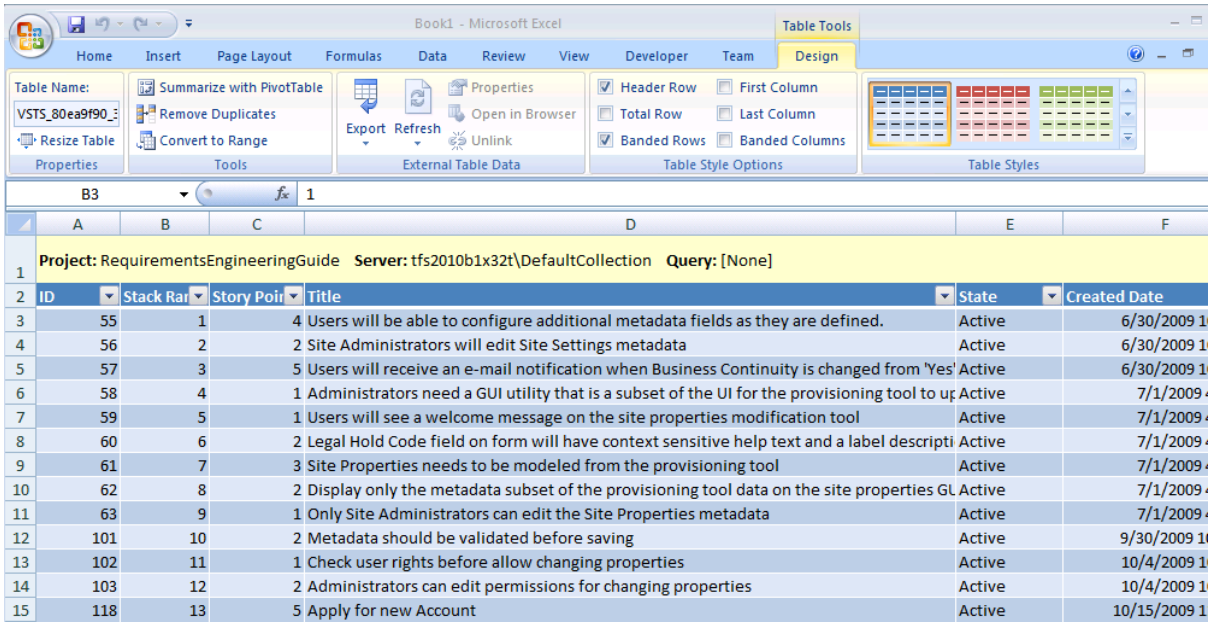
Each requirement type (represented as a work item type in Team Foundation Server) needs to be extracted from Team Foundation Server in an Excel spreadsheet. A view of all of the tractable attributes for the work item needs to be created in a work item query before taking the extract:



ID	Stack Rank	Stor...	Title	State	Created Date	Created By	Cl
55	1	4	Users will be able to configure additional metadata fields as they are defined.	Active	6/30/2009 10:12:37 ...	Administrator	10
56	2	2	Site Administrators will edit Site Settings metadata	Active	6/30/2009 10:12:37 ...	Administrator	10
57	3	5	Users will receive an e-mail notification when Business Continuity is changed from 'Yes' to '...	Active	6/30/2009 10:12:59 ...	Administrator	10
58	4	1	Administrators need a GUI utility that is a subset of the UI for the provisioning tool to upd...	Active	7/1/2009 4:51:35 AM	Administrator	10
59	5	1	Users will see a welcome message on the site properties modification tool	Active	7/1/2009 4:51:35 AM	Administrator	10
60	6	2	Legal Hold Code field on form will have context sensitive help text and a label description ...	Active	7/1/2009 4:51:35 AM	Administrator	10
61	7	3	Site Properties needs to be modeled from the provisioning tool	Active	7/1/2009 4:51:35 AM	Administrator	10
62	8	2	Display only the metadata subset of the provisioning tool data on the site properties GUI.	Active	7/1/2009 4:51:35 AM	Administrator	10
63	9	1	Only Site Administrators can edit the Site Properties metadata	Active	7/1/2009 4:51:35 AM	Administrator	10
101	10	2	Metadata should be validated before saving	Active	9/30/2009 10:55:58 ...	Administrator	10
102	11	1	Check user rights before allow changing properties	Active	10/4/2009 10:33:43 ...	Administrator	10
103	12	2	Administrators can edit permissions for changing properties	Active	10/4/2009 10:35:17 ...	Administrator	10
118	13	5	Apply for new Account	Active	10/15/2009 11:31:2...	Administrator	10

The picture shown above demonstrates a query that results in the Product Requirements and all of their applicable attributes. This will serve as the basis for the Product Requirements Baseline. A similar listing with the tasks for the project will be extracted as well and bundled with these.

The next step is to create an Excel extract of this list. The function to do this is boxed in red in the above picture. The result looks like the following:

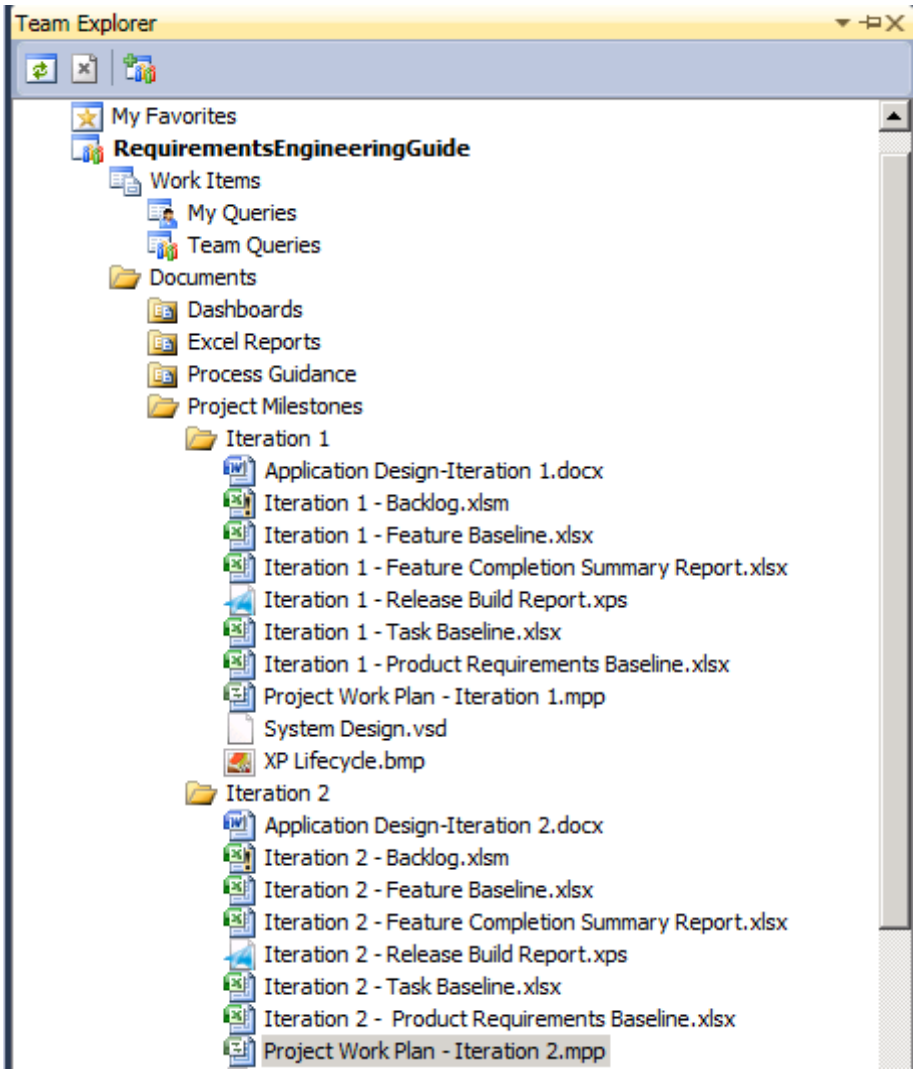


	A	B	C	D	E	F
1	**Project:** RequirementsEngineeringGuide **Server:** tfs2010b1x32t\DefaultCollection **Query:** [None]					
2	ID	Stack Rar	Story Poir	Title	State	Created Date
3	55	1	4	Users will be able to configure additional metadata fields as they are defined.	Active	6/30/2009 1
4	56	2	2	Site Administrators will edit Site Settings metadata	Active	6/30/2009 1
5	57	3	5	Users will receive an e-mail notification when Business Continuity is changed from 'Yes'	Active	6/30/2009 1
6	58	4	1	Administrators need a GUI utility that is a subset of the UI for the provisioning tool to up	Active	7/1/2009
7	59	5	1	Users will see a welcome message on the site properties modification tool	Active	7/1/2009
8	60	6	2	Legal Hold Code field on form will have context sensitive help text and a label descripti	Active	7/1/2009
9	61	7	3	Site Properties needs to be modeled from the provisioning tool	Active	7/1/2009
10	62	8	2	Display only the metadata subset of the provisioning tool data on the site properties GU	Active	7/1/2009
11	63	9	1	Only Site Administrators can edit the Site Properties metadata	Active	7/1/2009
12	101	10	2	Metadata should be validated before saving	Active	9/30/2009 1
13	102	11	1	Check user rights before allow changing properties	Active	10/4/2009 1
14	103	12	2	Administrators can edit permissions for changing properties	Active	10/4/2009 1
15	118	13	5	Apply for new Account	Active	10/15/2009 1

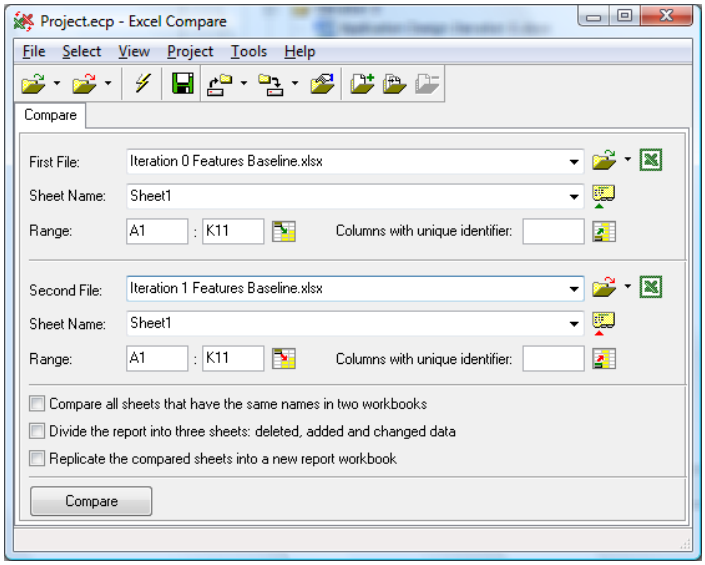
This spreadsheet needs to be saved. Develop a naming convention appropriate to the milestone. This one, for example might be labelled “Iteration 1 Product Requirements Baseline”. A complementary worksheet would be labelled “Iteration 1 Task Baseline”.

Store each of the files linked to the baseline in a project milestone folder in the documentation library created for the purpose of storing the baseline. ALL files associated with the requirements that are

targeted for the baseline should be stored in the same document library as the requirements baseline. Due to the nature of this baseline, this mechanism will ensure that they are all stored in SharePoint with a time stamp and the identification of the user that stored them there. If the organization requires governance compliance, such as FDA CFR-21, part 11, then a signature document can be created and stored with the package, or it can be bundled into a package that can be shipped to a facility that maintains digital signatures on the package.



If analysis needs to be performed to determine requirements volatility or just to understand the change from one milestone to the next, then a differencing tool, such as Excel Compare from Formula Software, Inc. (http://www.formulasoft.com), can compare the requirements baselines of a similar type from one milestone to the next. (i.e. Iteration 0 Feature Baseline.xlsx vs. Iteration 1 Feature Baseline.xlsx)



The comparison reports can also be stored in the Milestone directories as a source of difference from one iteration or milestone to the next. They also serve as a calculator for requirements volatility.

For a traditional project, controls are not as natural. Changes to requirements require a workflow that includes taking unplanned time during development to analyze the impact of the change, estimate the work effort, and then position and gain approval to make the change to the project. Larger, more established organizations will use change control systems to support their changes and approvals cycles. By using Team Foundation Server, a change request Work Item type can implement the change management workflow, where it can have a state-driven lifecycle that allows for the analysis, estimation, positioning, and funding approvals gated. Once approved, the team can then incorporate the change into their development effort. The change request work item can be linked using the same link mechanism described for traceability to the requirement or requirements that are added or updated on behalf of the request.

It would be important to compare the requirements set from one change to the next in order to measure “Requirements Volatility”, a measure used to indicate low maturity and quality of an application development team and its releases. The higher the volatility, the higher the likelihood that defects will be found late in the life-cycle or even after release. In the next section, we’ll describe how to leverage Team Foundation Server for supporting volatility or change control and approvals.

The process described above for comparing complete sets of requirements within baselines identifies differences at a macro level between baselines. When differences are found, further analysis at the document level can be made to drill into specific differences. Using Word, enabled with revision tracking and the version management capabilities of WSS, an analyst can compare and approve differences between individual versions of a document. The primary reason for this would be for “in flight” change management.

The figure below (Figure: 7) demonstrates a comparison of a word document to an earlier version of it. Note that the current revision is viewable in one window pane, the previous version in another and then the original document in a third. On the left is a view of the revision activities waiting for approval or rejection.

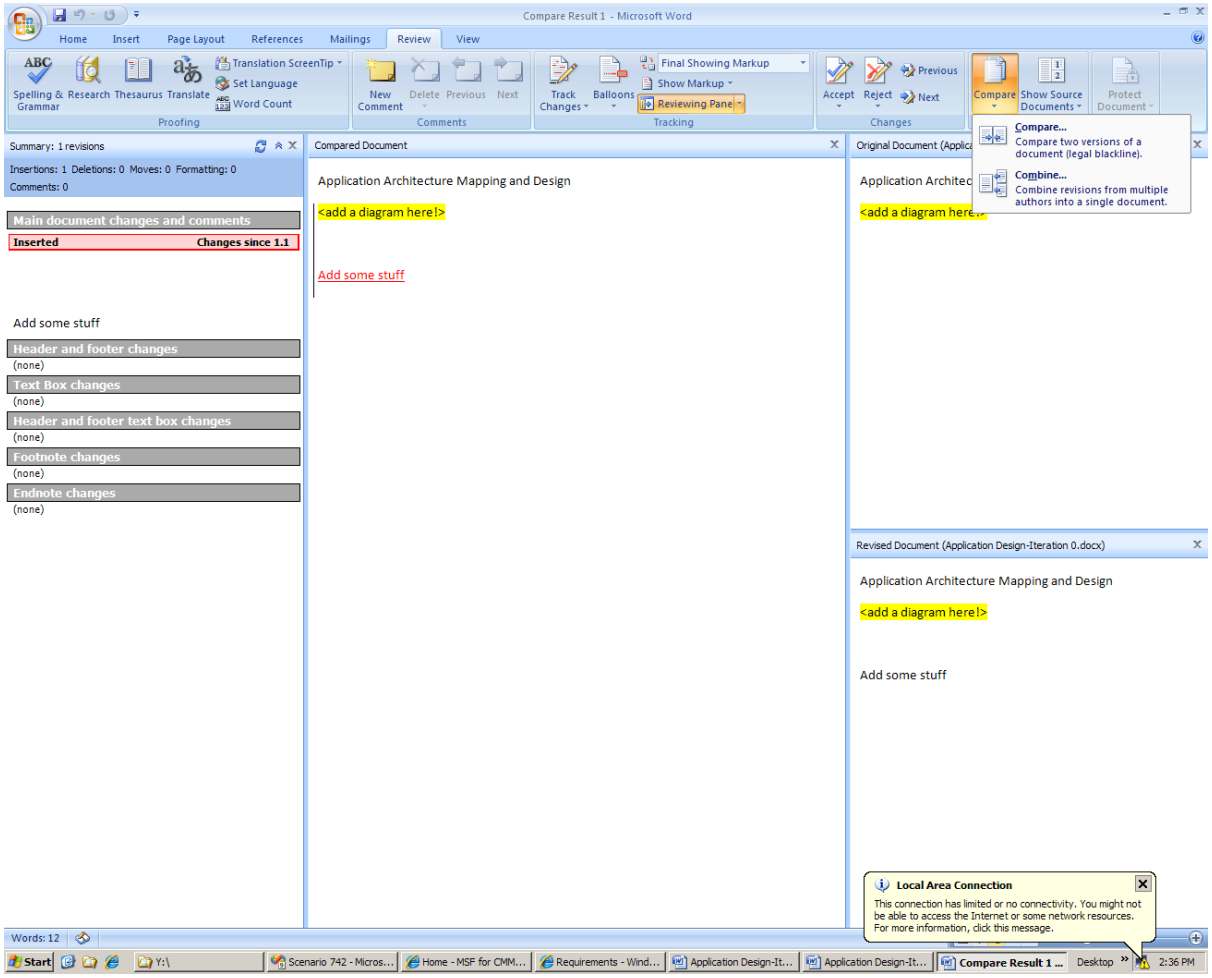


Figure 23: Comparing revisions of a Word document

Change Management Process

The flow chart below depicts the recommended process that should be followed for change requests when an agile development methodology is followed:

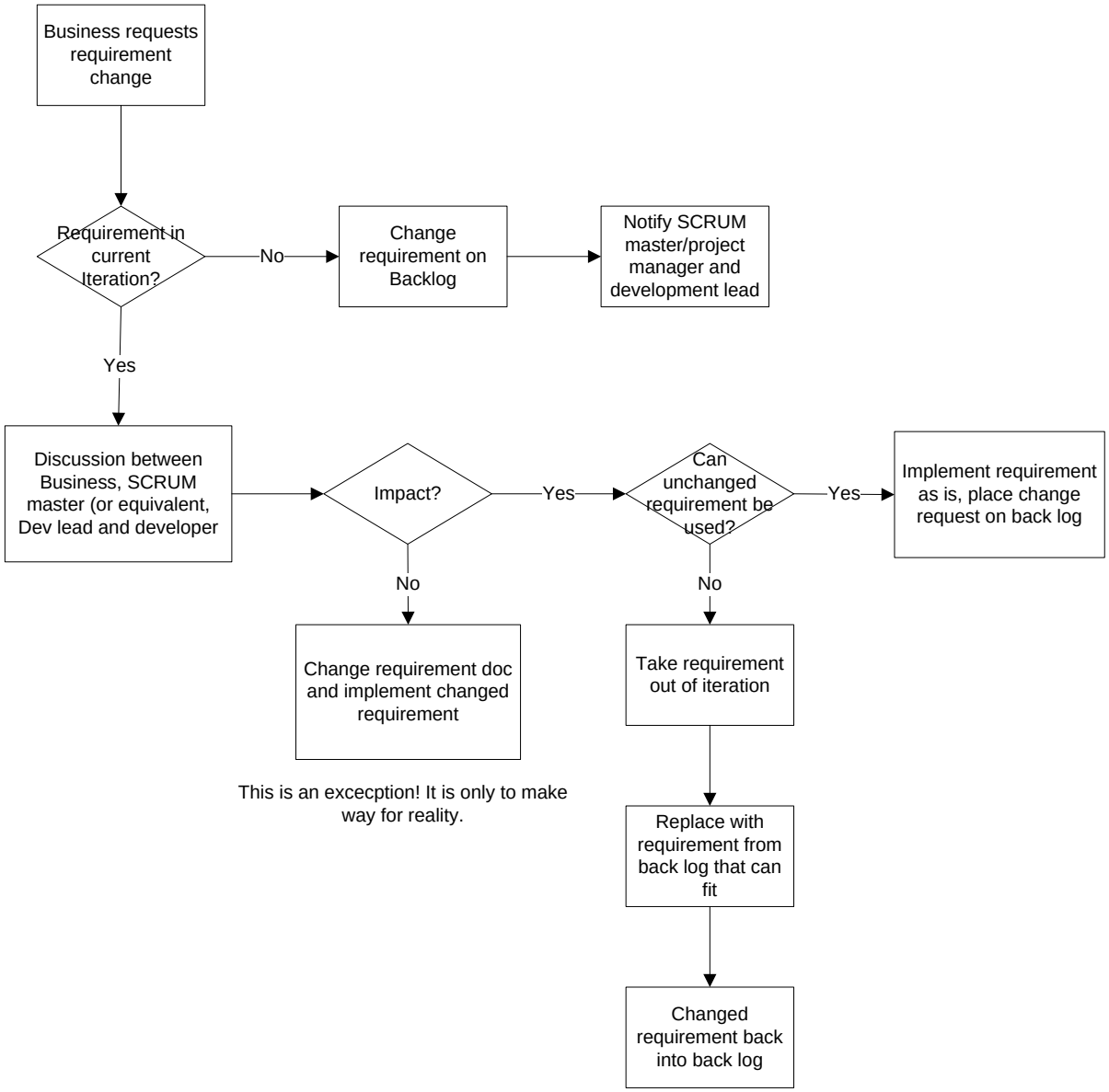


Figure 24 Process for change management for an agile project

Business requests a requirement change

When Business requests a requirement change, the person managing the requirements will follow a process to determine what actions to take.

A new requirement

If a new requirement is raised it needs to be analysed and defined appropriately. In agile methodologies requirements are captured in the form of user stories or scenarios.

A new requirement will simply be added to the backlog of requirements in the form of a scenario work item. A priority is provisionally assigned to the new work item. At the start of the next iteration the new requirement might be included in the iteration if the priority assigned to it is high enough.

Requirements destined for the next iterations will be approved by business just before the new iteration start. The new requirement will be treated as would any other requirement on the backlog.

A new requirement should never be accepted into the current development iteration or sprint. Should that requirement arise, the current iteration will be stopped and a new iteration started with a different set of deliverables. Since iterations should be fairly short (anything from two weeks to six weeks) very few new requirements should be urgent enough to necessitate the stopping of a current iteration and the start of a new one.

A change to a current requirement on the back log

Apply the change to the scenario work item on the back log. Any documentation attached to the work item should be versioned. This way history is kept of changes to requirements should questions arise at later stages.

The changed requirement will be on the back log and approved in its final form by the Business mere days before the start of the iteration that will deliver on the requirement. A changed requirement on the back log will be treated as any other requirement.

A requirement on the backlog will have an approval status of Draft and a minor version number.

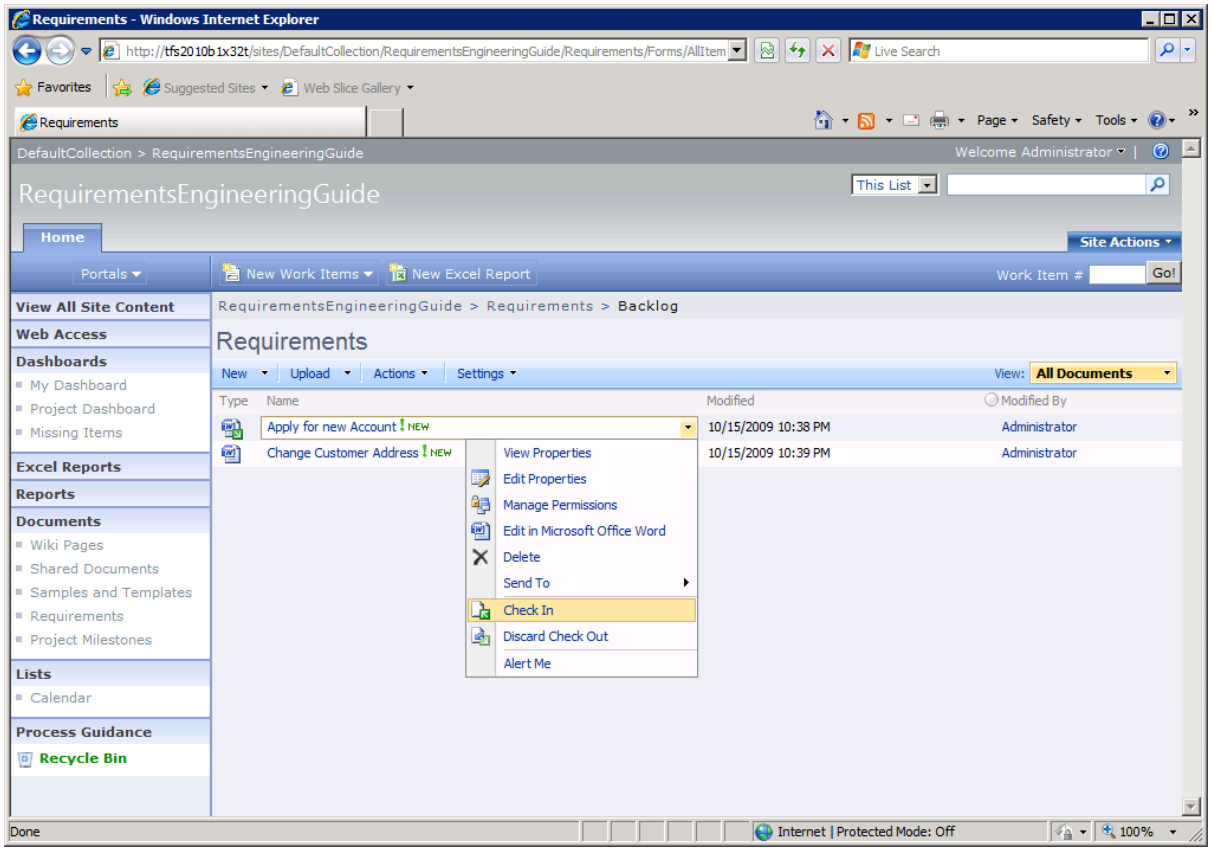


Figure 25 Requirement on the backlog

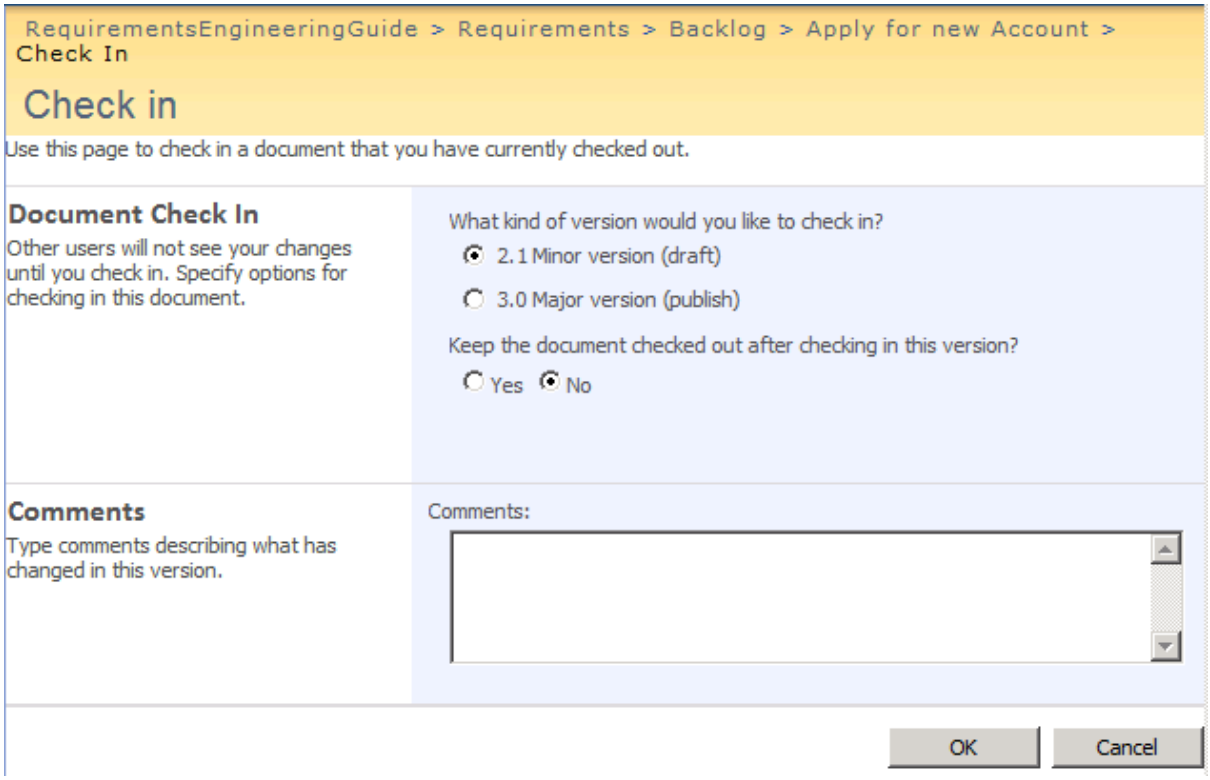


Figure 26 Check a document in as a minor version

A change to a current requirement being developed

In agile methodologies, change requests are normally not acceptable on requirements being delivered on in the current iteration. We know however that practicality often overrides theory.

If the change is to a requirement that is being developed, the change has to be discussed with the SCRUM master or project manager, the lead developer as well as with the developer to whom the work item is assigned.

The following should be determined:

Can this change request be seen merely as a clarification on the requirement?

A change can only be seen as a clarification on a requirement if it adds detail and not functionality. If it is merely a clarification, it can be implemented without hesitation and should not affect the development team's ability to deliver on the iteration. The scenario work item will be updated with the clarification and all related documentation in SharePoint will be updated and approved. As usual, a history of the changes will be kept on any documentation updated in SharePoint. A remark should be entered on the history tab of the scenario work item.

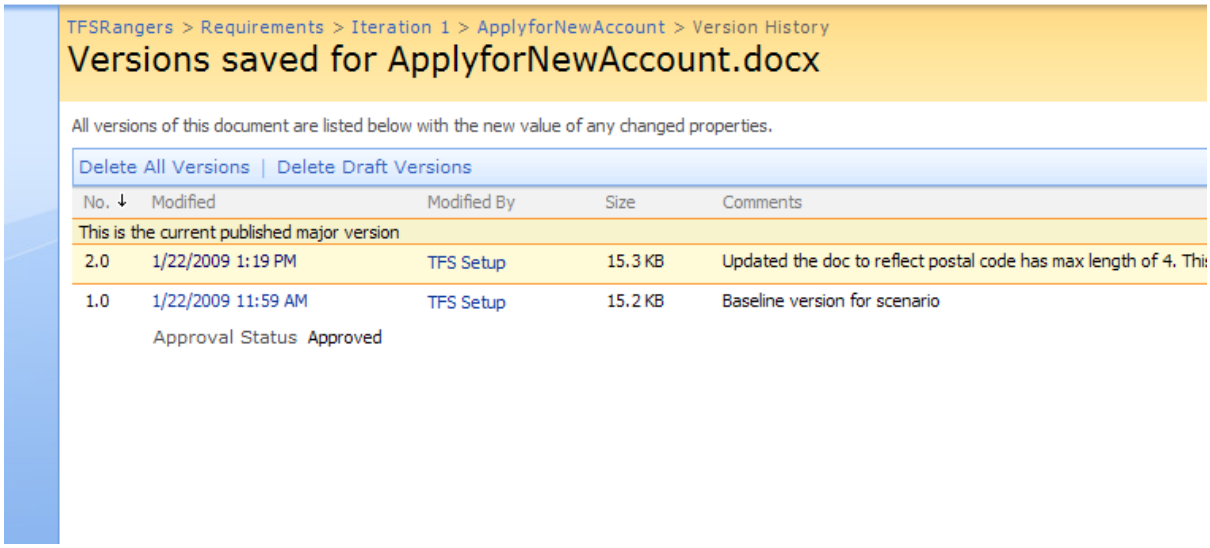


Figure 27 Document version history for a clarification of a requirement

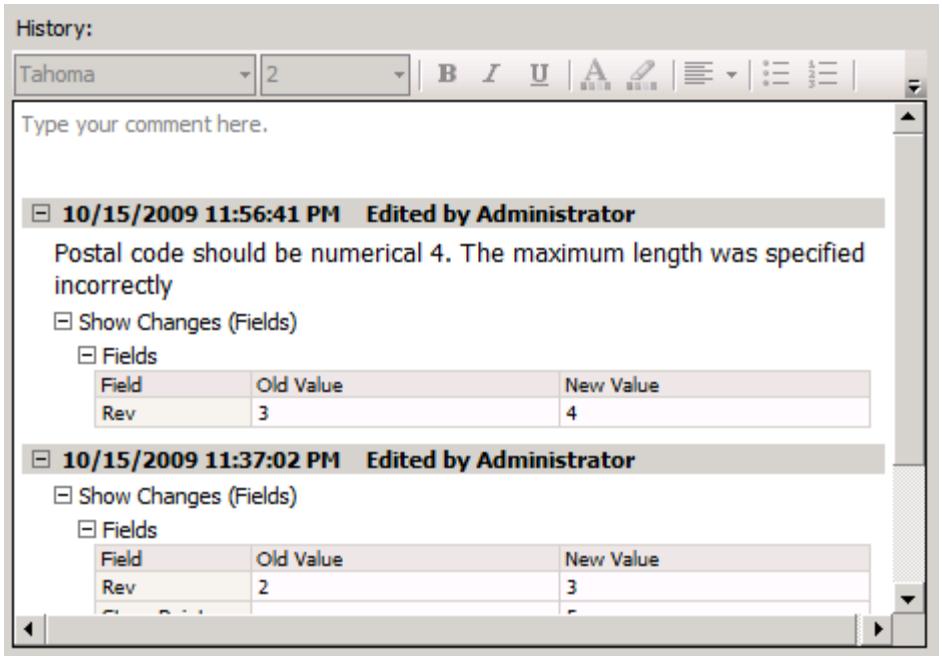


Figure 28 A note should be added to the history tab of the work item

Is the requirement useful without the change?

If the change to requirement is so significant that the requirement in its original (approved form!) is rendered useless without the change, the requirement should be taken out of the iteration and be placed back on the back log with the change request applied to the user story. This decision should be discussed clearly with the business. A change request should never impact on a team's ability to deliver on the iteration.

In order to remove a requirement from an iteration, change the iteration path of the work item back to the root and move the documentation on the team site to the Backlog folder. Tasks related to the work item should also be moved to the backlog.

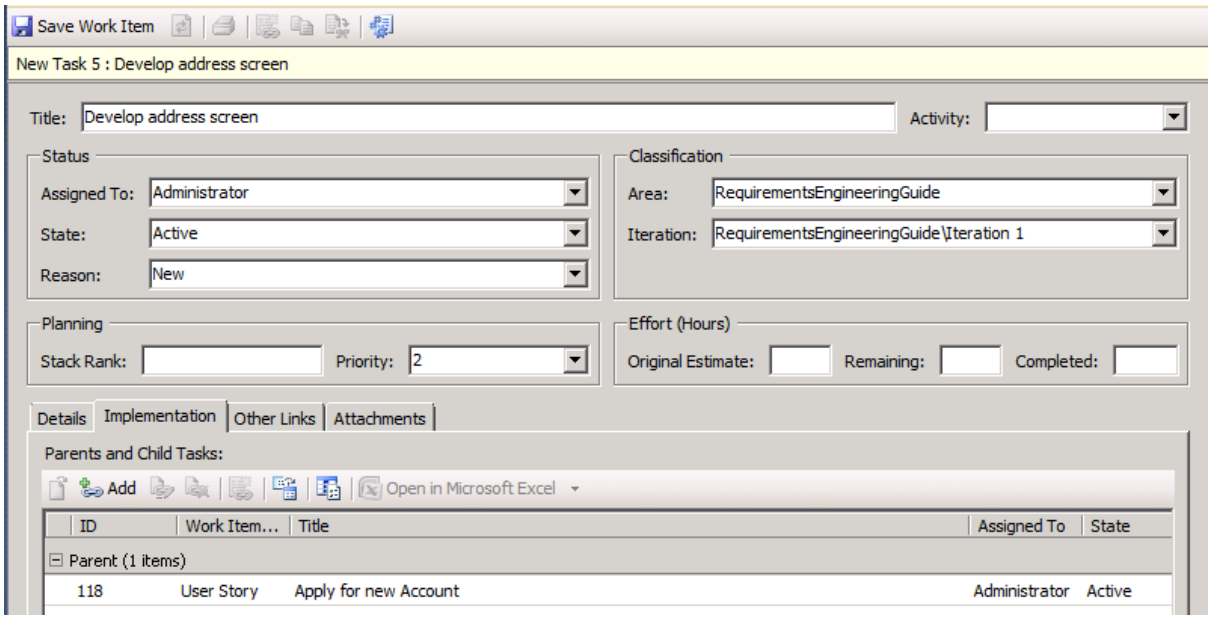


Figure 29 A task has been created to be completed in Iteration 1

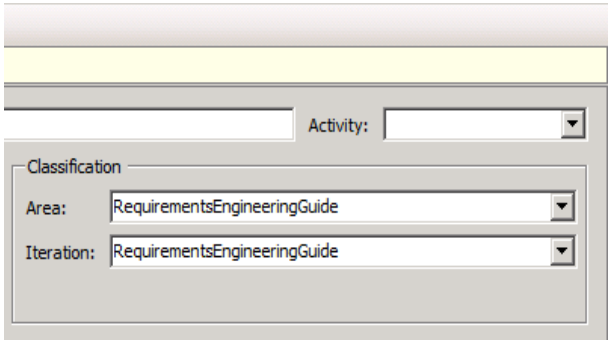


Figure 30 The task is no longer assigned and is on the backlog

Change Management Workflow Specific to a Traditional Project

For agile projects, a product backlog is prioritized and a manageable list of items is taken from the highest priority of the list and used for the upcoming iteration. Whether the requirements are perfectly understood or not, does not matter because the team will qualify the requirement as part of the iteration. For a traditional project, however, these ambiguities may force the team to need a change request. If the requirement can be used as is, a change request work item has to be created to provide an approval cycle to determine the change's impact to the project. The scenario with the Changes will be captured in the Change Request work item. The documentation attached to the User Story should be updated and versioned as usual. The developers should be informed explicitly not to use the updated

version during the current iteration. The updated version should also not be linked to the User Story work item under development.

When the changes are made, the scenario and all documentation should be updated immediately. The team should be explicitly informed of the change and the new version of the documentation to be used. A history note should be made to the scenario work item clearly indicating what the change is as well as who approved of the change.

In order for these changes to be effectively communicated to the team or manager of the team, notification can be turned on in WSS to notify a user or group of the change. Notifications on work items, particularly early in a project, may have a "spam" effect and will diminish the effect of being notified.

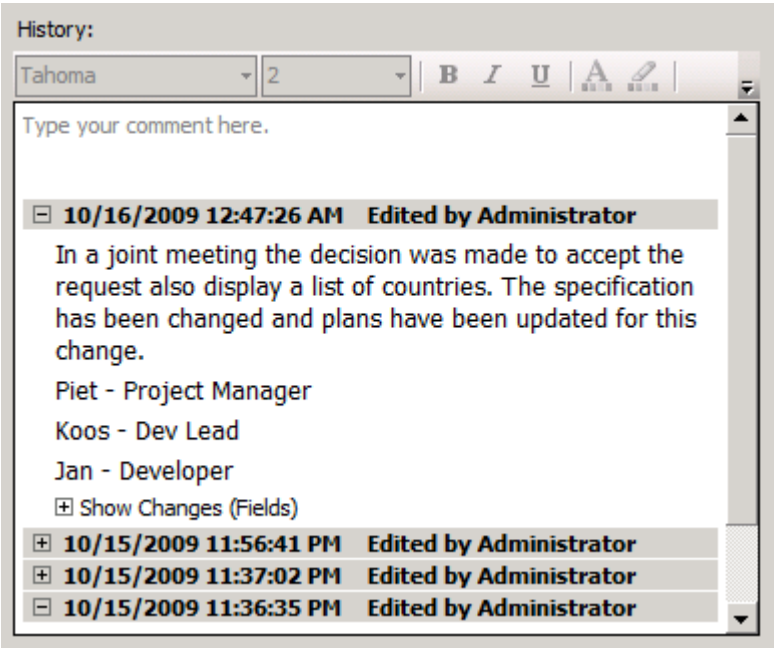


Figure 31 Clear history note for an emergency change is very important

The Approval Process

An approval process that is followed consistently is important even in agile projects.

The following process illustrates an approval process that is very well suited to agile projects as well as to more traditional SDLCs.
For an agile process this can be applied either in a "Pairing" situation informally between 2 team members, but will always need to be incorporated at the end of iteration on all artefacts completed for the iteration.
For a traditional project, this workflow provides the basis for a Change Request work item that represents a change to requirements that need to change or are "defective" in nature. The process can happen anytime during a traditional lifecycle and can also happen frequently.

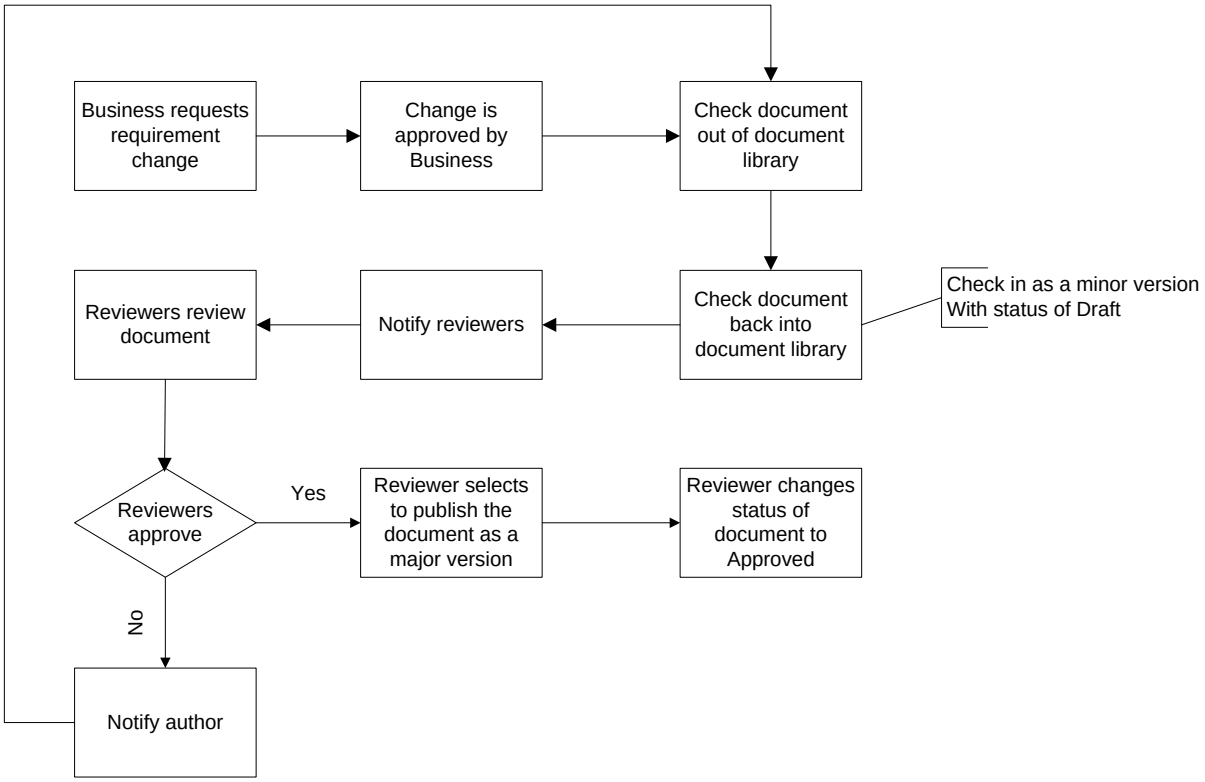


Figure 32 Lightweight approval process

Setting up a Team site to accommodate an approval process

When a Team Site is created for a new Team Project, the default setup is without version control and approval settings. It is important to change these settings if your team requires version control and approval. It is very strongly recommended that you always switch versioning as well as approval on to ensure you have a version history, a proper check out/check in process (as you would expect from your source control solution) and the sure knowledge that the documents you are working from are the approved versions.

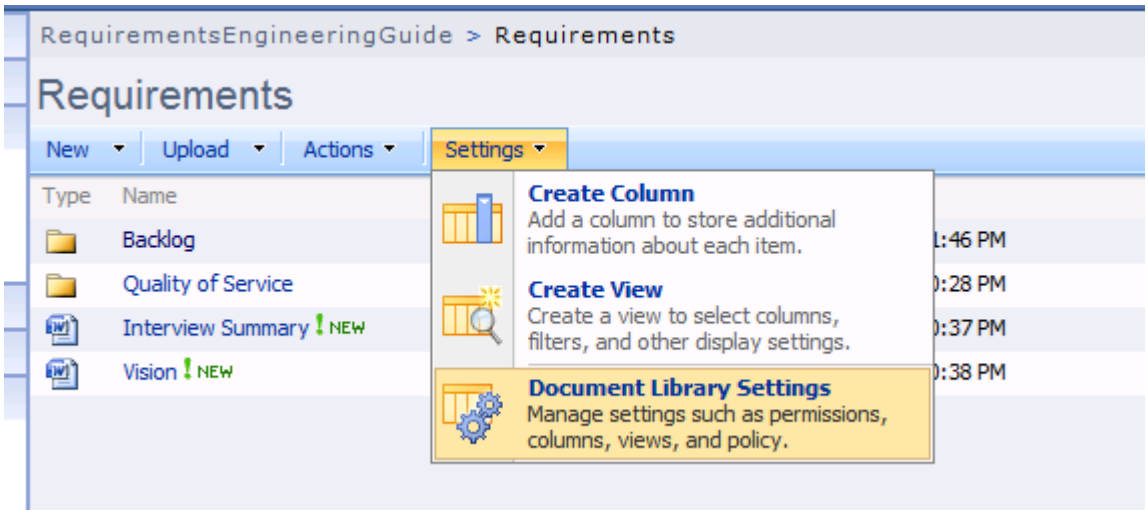


Figure 33 Where to enable versioning and approval

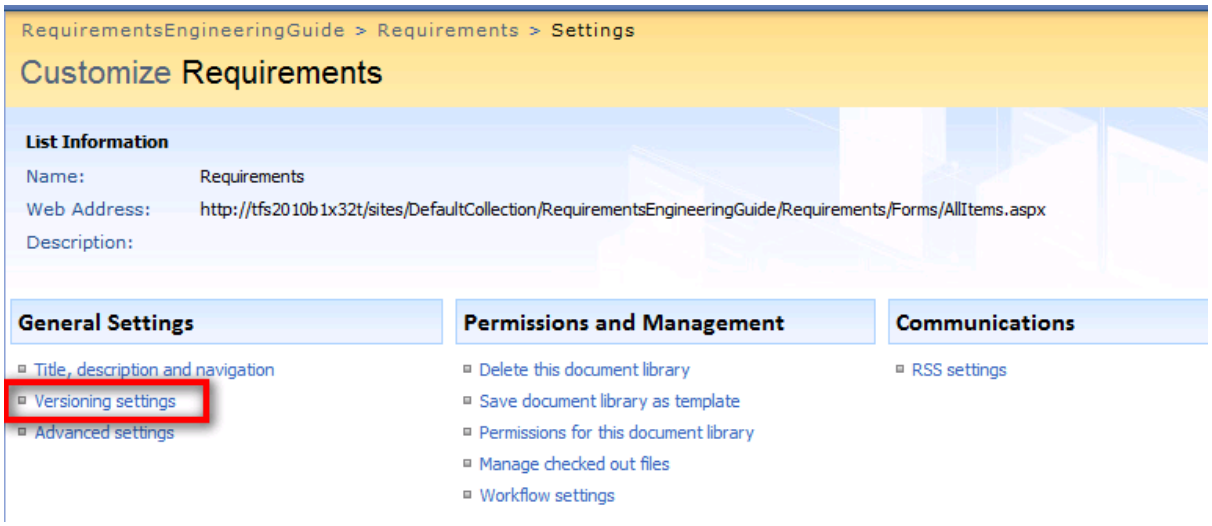


Figure 34 Customize Document Library setting

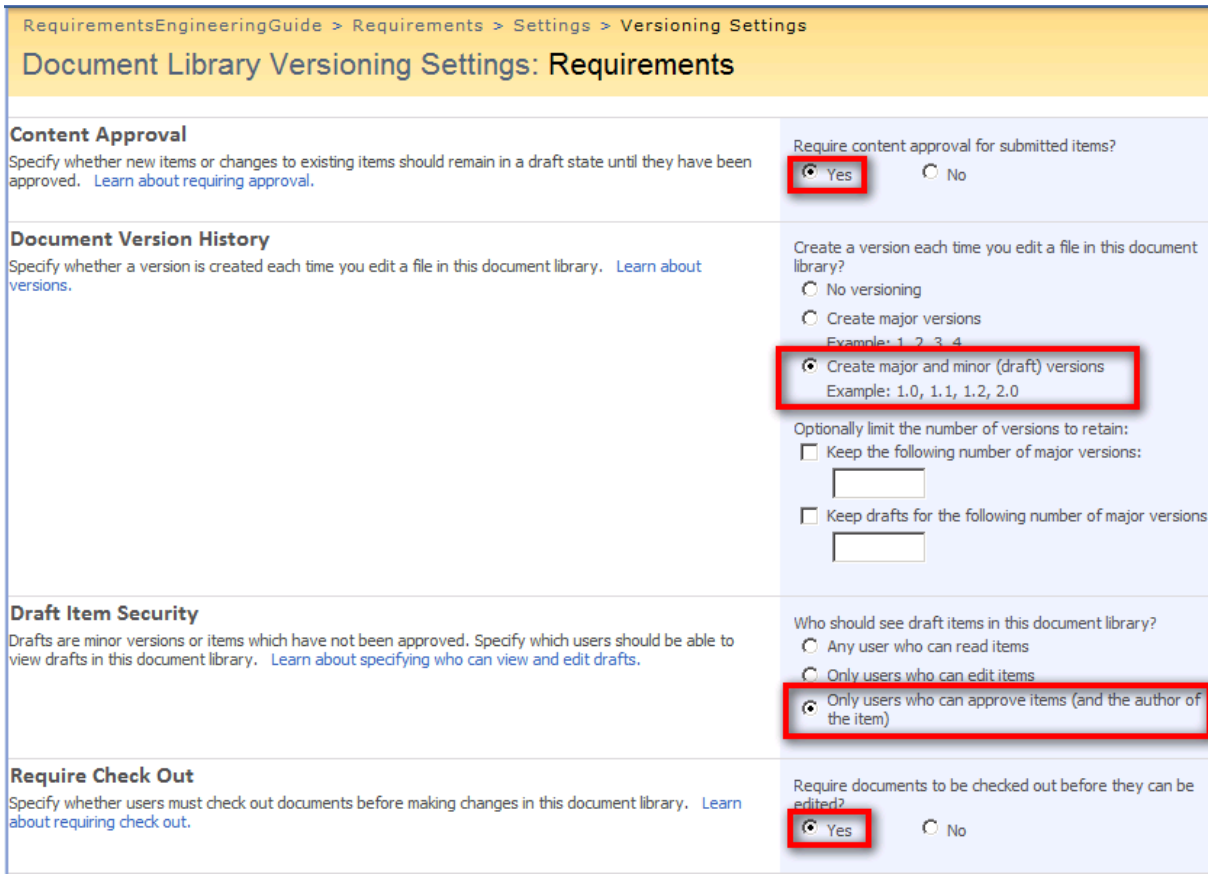


Figure 35 Recommended settings for Requirements document library

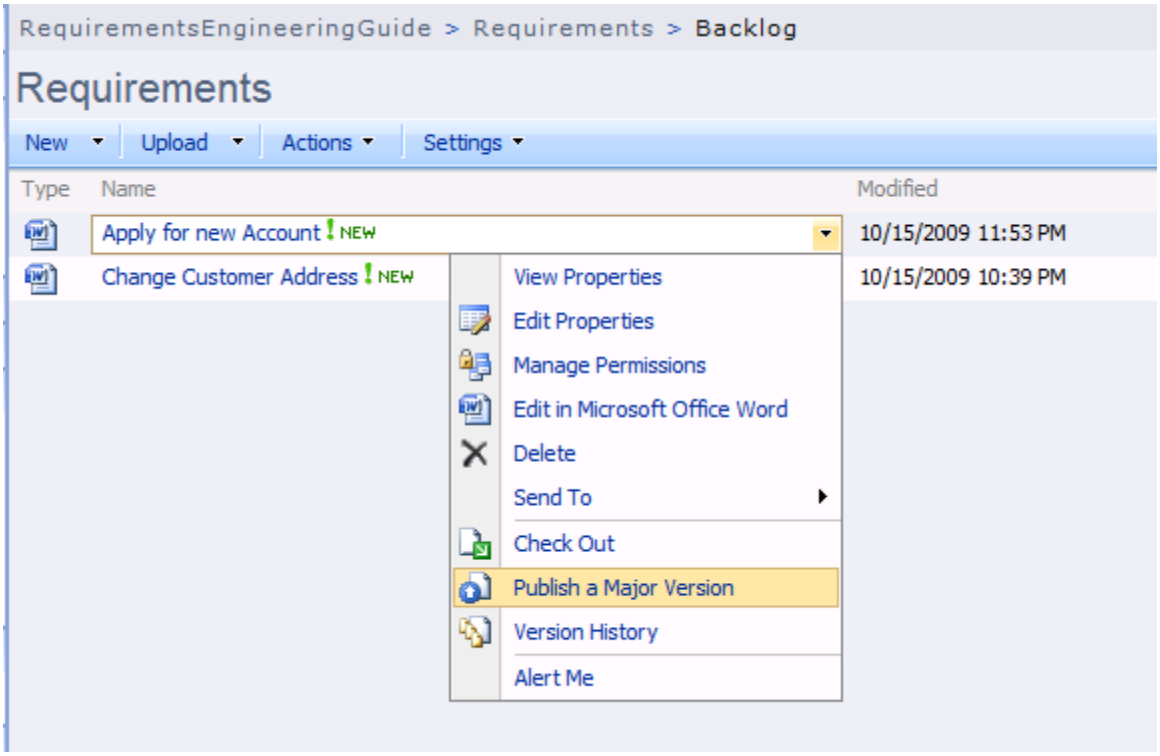


Figure 36 Document author publishes the document as a major version – approval status remains Draft

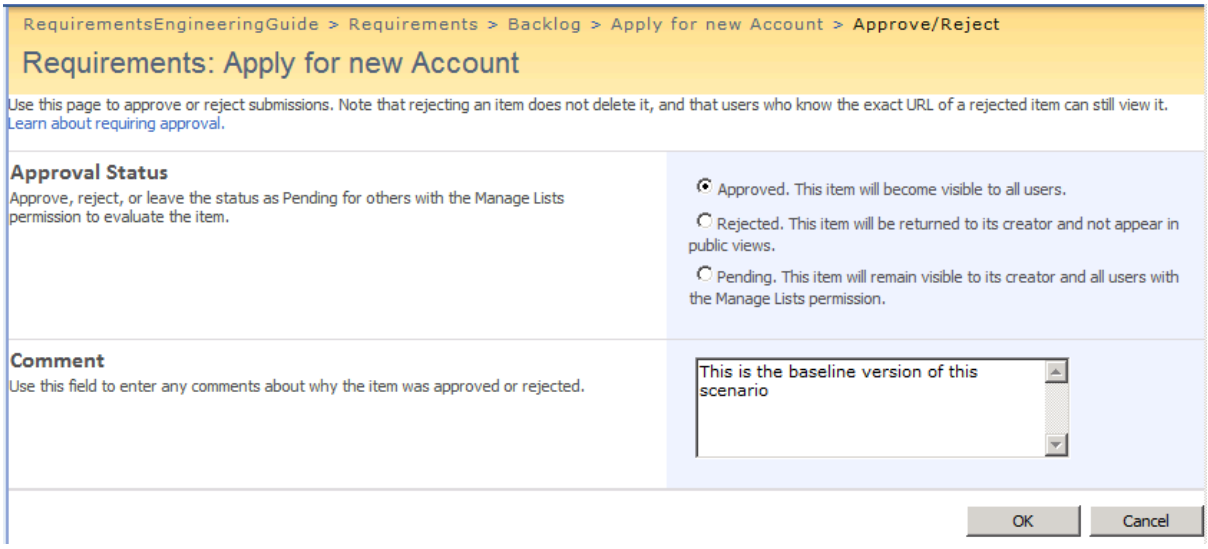


Figure 37 Approver approves of the document

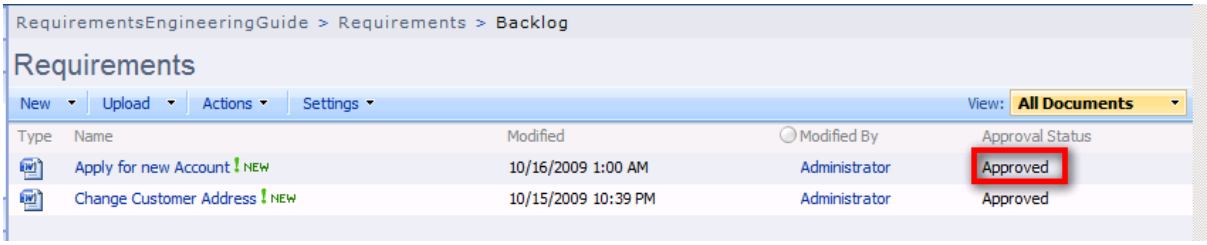


Figure 38 Approval status of the document

Final Thoughts on Change Management and Approval

Requirements change management practices are often overlooked in low process maturity organizations. Often times, even if the organization is mandated to submit to a formal change management process with a workflow technology, the changes made to all of the artifacts related to the request are not verified to be congruent with the desired change.

By following the guidance represented in this topic, teams leveraging Team Foundation Server as a repository for all project data will realize the benefits that make them more efficient, predictable, and robust. These benefits are just a few:

- Reduction in "Scope Creep"
- Improved Impact Analysis
- Comprehensive coverage of requirements tested
- Improved and effective team communication

Impact Analysis

"Change requests address not only new or changed requirements, but also failures and defects in the work products. Change requests are analyzed to determine the impact that the change will have on the work product, releated work products, and schedule and cost." – CMMI, Guidelines for Process Integration and Product Improvement, Chrissis, Konrad, Shrum.

Impact Analysis is performed to:

- Evaluate through activities that ensure that changes are consistent with all technical and project requirements
- Evaluate the impact of changes beyond immediate project or contract requirements (changes to an item used in multiple products can resolve an immediate issue while causing a problem in other applications)
- Determine impact that the change will have on the work product, related work products, and schedule and cost to the project

This topic area, though extremely important, is a topic area that is well covered in Change Control and Approvals as well as Requirements Traceability. As such, this section of the guidance will describe the high level scenarios of impact analysis and reference those sub-sections of the other topic areas for more specific detail.

> **Note:** The guidance for change impact analysis found in this section presumes that a business level requirement, implemented with a "Feature" work item in Team Foundation Server, is a given entity. The Rangers do not insist on using a "Feature" work item to accomplish this goal and do admit that it adds an element of formality that can easily be avoided on smaller and more agile teams.

Stories for Impact Analysis

Impact analysis workflow – Procedural guidance for analyzing impact to change

- Process for reviewing a change and its dependencies as well as documenting the resulting impacts.

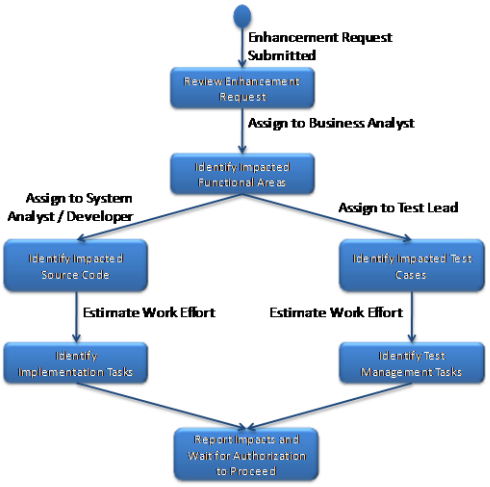
  When a change is introduced, it is either a request from the customer for something that was not in scope, or it was a misunderstood requirement that must be re-written to accommodate the misunderstanding. The former is an enhancement change request while the latter is a requirements defect. The same concept applies for design and test changes. They are, essentially, newly discovered facts that impact the scope and delivery of the project.

  For these changes, it is important that **Traceability** was maintained during the project and that **Change Control and Approval** workflows are implemented and enforced. Refer to those topic areas for specific detail to this guidance. Those topic areas describe Team Foundation Server usage for establishing and specifying traceability with links between work items and other assets as well as the change management, base lining, and difference comparisons required to understand the impacts of changes.

  Refer to the reports described in Requirements Traceability for support of the impact analysis activities.

Impact Analysis Process

At the point when a change is introduced, the development team needs to follow an evolutionary path through its requirements in a similar way to the process described in all of this guidance. The following diagram gives a high level view of that process:

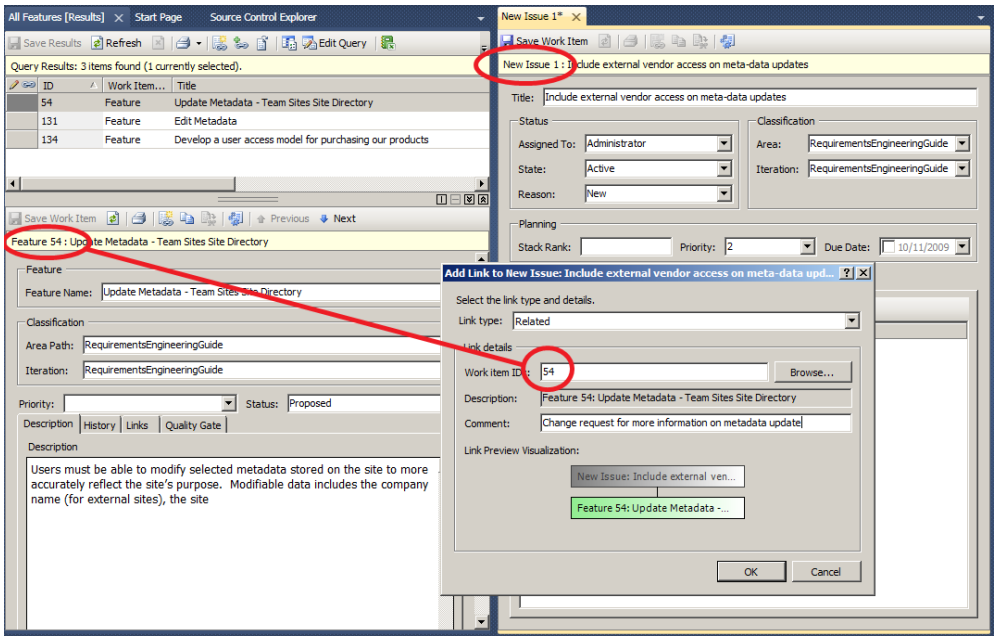


Review Enhancement Request

This activity is similar to business or scope analysis. The analyst's job is to understand exactly what the request means to the customer. Refer to the Analysis and Breakdown topic area for more detail. The focus will be on mapping the request to a Feature, User Story, or Requirement work item (depending on whether you're using MSF for Agile or MSF for CMMI) either by linking a CR (Issue/bug) work item to one, or creating a Feature, User Story, or Requirement to directly specify it.

Responsibility: Project Manager/Administrator

The following screen shot depicts a Feature work item (ID#54) that is in need of a change (New Issue). The Team Foundation Server user is in the middle of establishing the link between the Issue work item and the Feature work item.



Identify Impacted Functional Areas

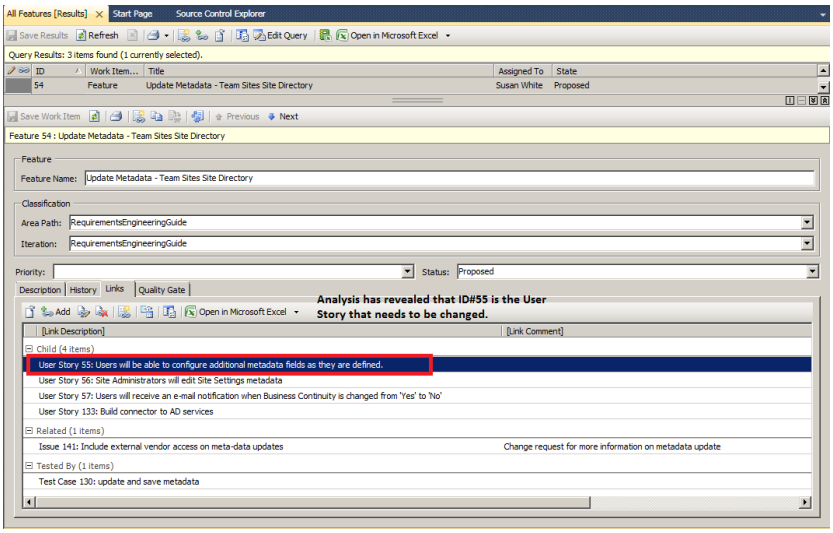
Using the correct elicitation techniques outlined in the Elicitation topic area, determine how to plan the analysis and gather all of the previous documentation about the system. If a project base lining process was followed as described in the Change Management and Approval topic area, the pre-existing documentation should already be easy to find and well scrubbed to allow efficient analysis. From that, the analyst will need to determine what needs to be changed, added,

or remove from the application to implement the change. This change can impact the solution in a few ways:

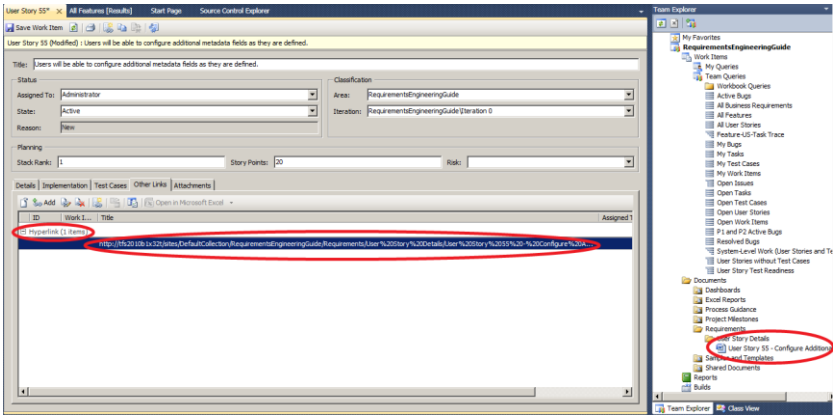
a. Change to add new functionality
b. Change to remove functionality
c. Change to change existing functionality
d. Addition of a non-functional quality of service (i.e. performance, reliability, support, operations)

If the change is to add functionality, then a new requirement and traceability for this request do not exist and needs to be created. Then, for all of the changes, manually identify the previously implemented work items (Requirements) that reflect similar desired functionality and determine how to augment it with the new capability. If the work items exist and their detail is stored in documents in WSS, then perform the following procedure:

- Begin by generating a query of all the functional requirements.
- Scan the list for candidate requirements; those that appear similar to the new functionality and Isolate them. The simplest way would be to review the user story requirements that implement the feature to determine if the change request affects them. The following graphic is of the features linked work items with a user story highlighted (#55) that appears to be impacted by the change.



- One candidate at a time, open its linked documentation or detail that, if following the trace model defined in the Traceability topic area, should be in the SharePoint portal of the baselined team project. Identify locations in the documentation that need changes.



- For each of the functional changes, create a Requirement work item and link to the document from which it was found using the URL link type. Each new requirement or user story work item will have a link to the same document shown above and possibly other documents in the same document library in the SharePoint portal.

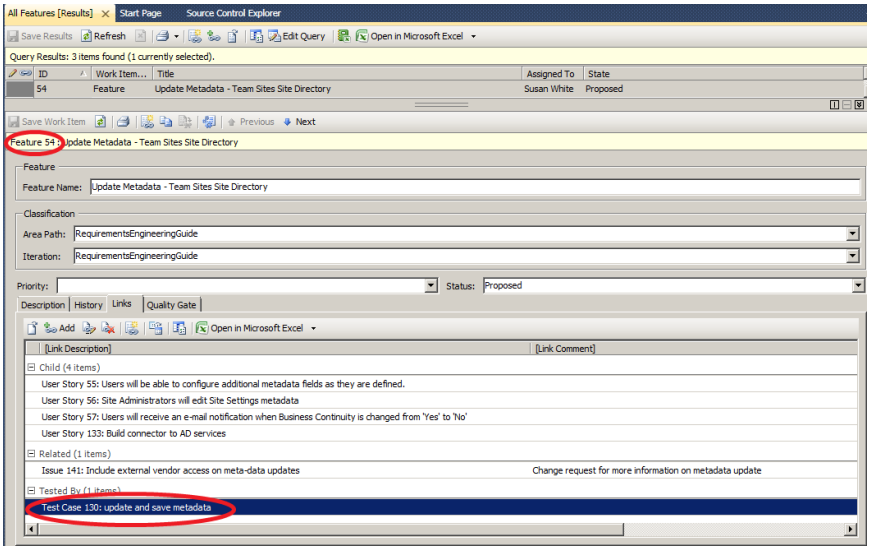
Responsibility: Business Analyst

Identify Impacted Test Cases

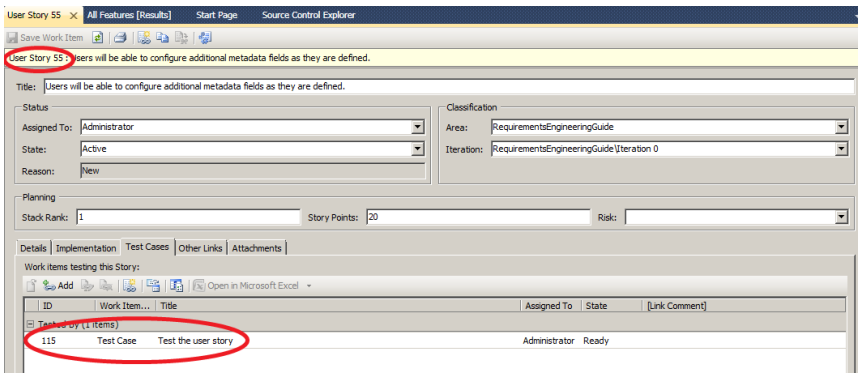
Using the traceability of the existing requirements, make a listing of the test cases that are linked to them. These test cases represent the highest priority regression tests that will need to be run during any test cycles. For the new requirements identified for the change, additional test cases need to be created and linked to the requirement using the Test Link-type.

Responsibility: Test Lead

Below is a feature with its user acceptance test that should be regressed:



And the following graphic displays the impacted user story (#55) and its system test cases that should be regressed:

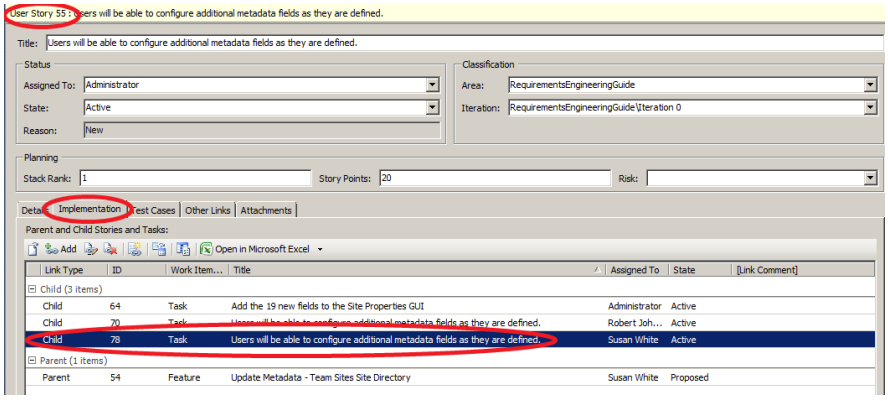


Identify Impacted Source Code

Using the traceability of the identified requirements, list the children tasks that were tracked to implement the solution. From the tasks, track to the change sets that were checked in and linked to them. The source code listings from those change sets represent the code that will need to change to implement the change.

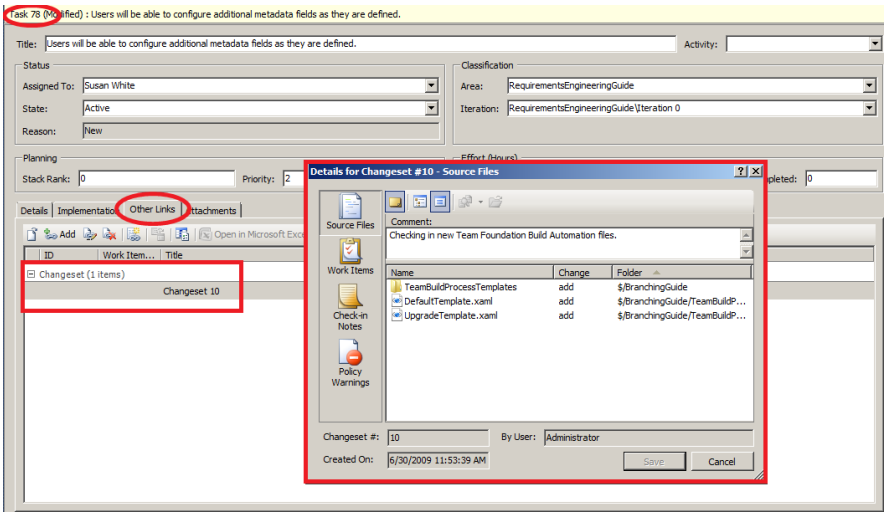
Responsibility: Developer / Architect

In the following screen graphic, we are highlighting the affected user story and showing its "Implementation" links. Implementation links are tasks that need to be accomplished in order to implement the requirement or user story.



In this case, the tasks would have been done already, but, with proper traceability, their change sets will reveal the source files that would need to be updated to make the new change.

The next screen shows that task (#78) with its change set open revealing those changed source files.

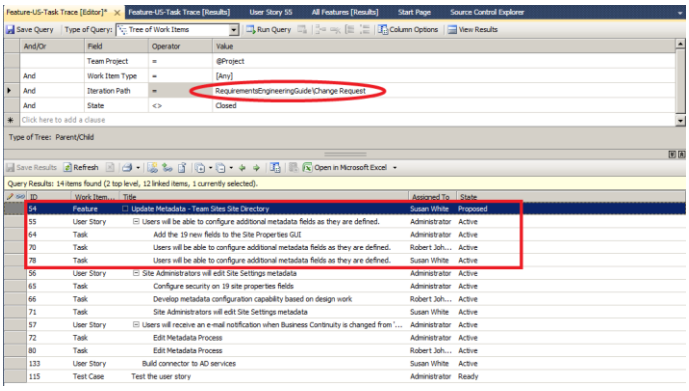


Estimate the Work

This activity is performed in two ways:

a. Identify Implementation Tasks – The developer and architect will analyze the new or changed functionality and identify new tasks required to implement the changes. The tasks need to be linked using the Implementation Link-Type (parent/child hierarchy)

The graphic below shows an ad-hoc query where the feature that needs to change has been allocated to a "change request" iteration. Using the tree-view query type, we are able to identify all of the requirements that implement that feature as candidates for that change. As we saw in the earlier impact analysis, we know that only User Story #55 is impacted and we've highlighted it and its implementation tasks in the graphic. New tasks will need to be written to implement the change, while the old completed tasks will identify the source files that need to be changed.



b. Identify Test Management Tasks – For each new requirement, identify new test cases that will verify the new requirements. Refer to the Validation topic area for ideas of how to be comprehensive with positive and negative test cases. Link the new test cases using the Test Link-Type. For each existing requirement that needs to be changed, review their linked test cases to ensure that their implementation reflects the changes. Once the test

cases have been identified, identify the task work items that represent the work required to implement the necessary changes to the test model. These tasks will be estimated and tracked during the change implementation.

Report the Impacts and Wait for Authorization to Proceed

At this point, all the necessary impacts have been identified. Gather the work items and generate a report that reflects the necessary work. The recommended way to do this is to tag each new and existing work item that were found with either an "Area" attribute value or common "Iteration" that represent the effort to make the change. From this, a report should be easy to create that filters all work items based on that value. If using iterative development practices, an "Area" should be used to identify all affected work items by a common change request while leaving the rest of the iteration open for new work on that iteration, otherwise a complete iteration for the entire change would suffice.

Final Comments on Impact Analysis

Requirements Change Impact Analysis is highly dependent on the structure and maintenance of the requirements and their established traceability. Without it, impact analysis can take an order of magnitude longer to perform than with it. The process outlined in this document is one of many that can be followed to identify and fulfill the work necessary to implement the desired changes.

3rd Party Integrations

Many software vendors and Microsoft partners have developed integrations with Visual Studio that provide capabilities that are not well implemented by Visual Studio Team Suite. Specifically, they have built capabilities in the following elicitation, specification, validation, and change management domains. Here is a list of some of the more tightly integrated partners.

TeamSpec

TeamSpec is a vendor that has built an integration between Microsoft Word and VisualStudio. Team spec enables requirements specification in Team Foundation Server with its integration that allows organizations to synchronize headers and paragraphs in Word with work items in Visual Studio.

For more information:

TeamSpec by TeamSolutions

http://www.teamsystemsolutions.com/teamspec

Joe Buys – jbuys@personifydesign.com

stpSoft

stpSoft has built a similar integration, but has also provided a Visio to VisualStudio integration that links use case and activity diagrams with work items in VisualStudio. Their integration specifically provides storyboarding as an elicitation technique.

For more information:

stpSoft

http://www.stpsoft.co.uk/stpbadeveloper1.html

Badr Khan – badrk@stpsoft.co.uk

Ravenflow

Ravenflow provides an application that supports requirements elicitation, specification, and validation. With its integration to Visual Studio, development teams can jumpstart their development with requirements that are represented as a proxy to Raven.

For more information:

Ravenflow

1900 Powell Street, Suite 500

Emeryvill, CA 94608

510-285-4600

http://www.ravenflow.com

IBM DOORs

DOORs is a rich requirements management environment that provides for elicitation, specification, change management and approval of requirements. Its integration was built for bi-directional synchronization with Visual Studio.

For more information:

Jared Pulham – jared.pulham@uk.ibm.com

Appendix

The following documents, process template elements, reports, checklists, and templates were referenced throughout this guide and are available in a separate bundle that accompanies this guide.

Requirements Engineering - Checklists.xlsx – Checklists used for validation, elicitation, approvals and change control

Document Templates Samples.zip – Word document and spreadsheets that can be stored in a Team Foundation Server document library for use on your projects.

Bibliography

ChrKang92 - Christel, M. G., & Kang, K. C. (1992). *Issues in Requirements Elicitation.* Pittsburgh, PA: Software Engineering Institute.

Rat99 - Rational Software Corporation. (1999). Guideline: Brainstorming and Idea Reduction. *The Rational Unified Process*. Rational Software Corporation.

SEI91 - Software Engineering Institute. (1991). Requirements Engineering and Analysis Workshop Proceedings, Technical Report. *Software Engineering Institute Requirements Engineering Project.* Pittsburgh, PA: Software Engineering Institute.