Cognizant Technology Solutions

Implementing **Cognizant Feature-Driven Development** Using Microsoft Visual Studio Team System

Technology White-paper

Cognizant .NET Center of Excellence
Bangalore, India
For further details, contact: Vincent.TP@cognizant.com

Table of Contents

Overview

This document introduces Cognizant's Feature-Driven Development (FDD) process and explains how it can be implemented using the Visual Studio Team System (VSTS) templates. The concepts explained here also demonstrate how any custom software development process can be automated using VSTS.

However, the implementation of Cognizant FDD templates is still under development and may differ in the final version from the way it is described in this document. The implementation of Cognizant FDD template is done on VSTS Beta1 interim release (March 2005).

Introduction to Agile Methodologies

Software engineering methodologies entail a disciplined process in order to make software development more predictable and efficient. This is accomplished by developing a detailed process with a strong emphasis on planning inspired by other engineering disciplines. The most frequent criticism of these methodologies is that they are bureaucratic. Often, these methodologies are very elaborate with too many stages in between that slow down the pace of development. Also, these well thought-out, top-down approaches fail to react to frequent changes in requirement, which are a business reality. Alternative views, such as having no formal process except good programming, are also equally risky. Such approach suffers from many more disadvantages such as excessive chaos, inability to scale beyond a few team members, lack of repeatability, lack of easy transferability to the other teams, and so on.

In the last few years, several new methodologies have appeared that are midway between these two extreme viewpoints. For a while, these were known as lightweight methodologies. Now, the accepted term is agile methodologies. The agile methods attempt a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff. The most immediate difference is that they are less document-oriented, usually emphasizing on a smaller amount of documentation for a given task.

Agile methods are adaptive rather than predictive. Engineering methods tend to try to plan out a large part of the software process in great detail for a long span of time. This works well until things change. So their nature is to resist change. The agile methods, however, accept change as normal. These methodologies tend to be processes that adapt and thrive on change, even to the point of changing themselves. Agile methods are people-oriented rather than process-oriented. Agile methods assert that no process will ever make up the skill of the development team, so the role of a process is to support the development team in its work. Several methodologies come under the agile umbrella. Feature-Driven Development (FDD) is one such agile methodology.

While considering any process, it is important to note the unique characteristics of the type of software development. Broadly, there are two categories of software development - custom business software projects executed by software services companies/system integrators and the software products built by product companies. Product companies have in general much more control on the technology choices and relatively less number of stakeholders. In contrast, the business software that is built by a software services organization for another client organization tend to have more stakeholders and are often based on the latest technology trends in the market, current IT assets and investments, technology and tool biases, available skills and some specific factors in the client organization. Invariably, the requirements conceived and owned by one organization need to be communicated and translated to implementation by another organization within a very short time period. Most often these requirements undergo a lot of changes since the work involves communication between teams from two different organizations. The architecture and design will involve reuse of many prefabricated components from within and from other 3^{d}-parties to reduce the effort and time for development (hence the term “system integrator”). Most of these projects are handled in extremely compressed timelines. The methodology explained in this white paper is more focused on the lifecycle of business software projects rather than product development. At some places, this difference may become very evident. For example, in a typical product development, it is quite common for the programmer to know about the functionalities and even own the design. Large-scale business applications in a completely new domain need more granular roles and skills. While applying the same principles of agility, these key differences between products and projects need to be considered.

Overview of Cognizant FDD

FDD is a model-driven short iteration process, which is designed and proven to deliver frequent tangible client-valued functionality repeatedly with emphasis on quality. Client here refers to the end user of the software. It also enables accurate and meaningful progress-tracking and status information with minimum overhead and disruption for Developers in a project. It is a proven agile methodology that can scale well, handle changes efficiently, and is lighter than the traditional heavyweight processes.

FDD uses small, client-valued functions called "features", which are tiny building blocks for planning, reporting and tracking. The end-to-end implementation duration of these features will last about two or three weeks. The process facilitates inspection, enables concurrent development, and tracks and reports progress with efficiency and accuracy.

An overview of the FDD process is as follows.

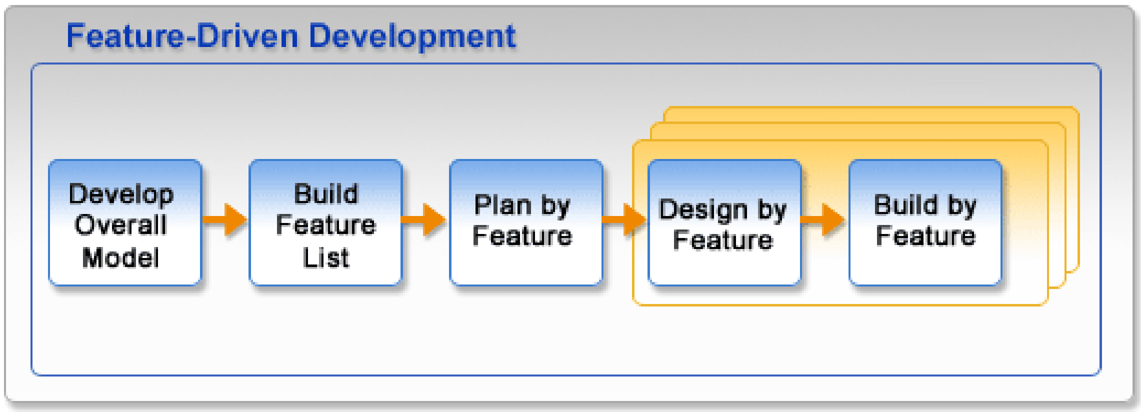


Figure 1: Traditional Feature-Driven Development

The important step is to split the requirements as features, which are then implemented in an extremely iterative manner. The requirements are broken down into features. Features are explained in client-valued functions with a clear syntax like:
<<Action>><<result>><<object>>

The following is a good example of a feature.

Calculate penalty amount for late submission.

The domain model defined at the beginning of these feature-based iterations serves as a solid foundation. Based on the requirements translated to features, they are prioritized based on business needs and dependencies. Based on this prioritization, a single feature or a group of features (also referred to as feature-set) is taken for implementation. Implementation follows a domain walkthrough, design by feature and then construction and testing by feature. The most significant difference between this approach and the other methodologies is that the design and the build happen at the lowermost granularity of a client-valued function.

Cognizant has implemented a number of large-scale business applications using the FDD model. Based on our experience and learning, we have extended the traditional FDD process to handle large-scale application development. The need for extension for large-scale application development is based on the following considerations.

- The design and implementation of a feature should not upset the overall structure of the system built for implementing previous features. This agility in large applications can be ensured only by having in place a low fidelity design of most of the critical components before the implementation begins.

- Components can be based on the domain model. But more often than not, the physical components in a system will be guided by a number of other considerations such as layering, tiers, reuse of components, usage of 3d-party products and will also be optimized for a number of other aspects such as performance and scalability. So, we need to have a good definition of components and frameworks before we can start the feature-driven development. Also, the component model of the entire application should remain more or less stable throughout the development. Otherwise, the implementation of a feature can upset the structure built for the other predecessor features and the whole system structure can become unstable.

- FDD needs to align with relatively well-accepted standard roles in software industry such as Architect, Designer, Developer, etc. For example, the artifacts of the analysis of a model of the problem domain and the design model of the solution domain need a bridge. Also, classic FDD has limited roles for functional expertise. One or two domain experts doing a domain walkthrough and passing the information to technical team may not be really feasible when the application functionality involves complex business processes. (Most of the FDD story is written by product companies. It may not be the ideal comparison to the

Cognizant Technology Solutions

development of a complex business solution by system integrators.) In case of a developer, understanding the functionality of an editor or a modeling tool is quite different from trying to understand insurance business and associated business rules. Probably, a more ideal, scalable and cost-effective model for business application development may be one where the Developer need not understand the business.

- For solution integrators, the development process often needs to support integration or modification of many existing legacy systems that may not be based on an ideal domain or component model.

In order to ensure that these aspects of real life business application development are addressed, the traditional FDD is extended to form the Cognizant FDD.

Cognizant FDD takes some of the good concepts from RUP, such as architecture-centric development and component orientation that are essential to establish the stable foundation and applies core agility principles of FDD.

A few salient features of Cognizant FDD are:

- System definition phase is identified as a key phase for Cognizant FDD. In this phase:
  - The domain model is created as per classic FDD.
  - The domain model is just one of the inputs to the overall definition.
  - Complete SOA-based architecture is defined.
  - The service definition is done by top-down analysis and bottom-up analysis (for leveraging the existing IT assets).
  - Low-fidelity component definition is done based on domain model and also on various other considerations such as layering, reuse, deployment, performance, etc.
  - Framework definitions are done during this phase. (Reuse of prefabricated components and frameworks such as Microsoft application blocks or internal frameworks such as Cognizant's CAFÉ.NET)
  - A spike test is carried out to ensure that non-functional requirements such as response time, scalability, security, etc. meet the expectations.
  - Prior to feature-wise development, it is ensured that the base structure to incrementally deliver the features is available.
- The standard FDD process is used to support the subsequent development phases.

- This also includes a few modifications to address the functional complexity.
  - Cognizant FDD includes additional workflows for change management and bugs.
  - Cognizant FDD includes additional roles to align with large-scale development.
  - Cognizant FDD includes additional support processes for ensuring CMM Level-5-compliance.

The following provides an overview of the Cognizant FDD.

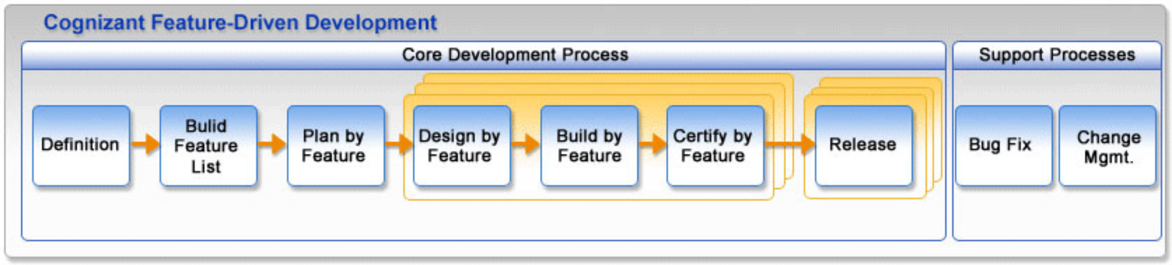


Figure 2: Cognizant Feature-Driven Development

At a high level, this process flow is almost like classic FDD. The only difference is that actual physical components are created during the definition instead of the domain model phase.

A more elaborate view of the process as follows will highlight more details and the key differences.

The development process engine takes end user requirements and delivers it as client-valued functions or features in an extremely iterative fashion. The overall development process is split into 3 areas: functional, technical and managerial. The functional or the feature area is all about business, domain, functional requirement, analysis, etc., in short, everything related to end user or the “what” part. The technical side is all about the “how” part, such as non-functional requirements, architecture, design, construction, etc. These two streams need teams with different skills and both need to approach the same issue with different perspectives at varying degrees of details across the implementation lifecycle. The management ties up these two streams with additional activities such as planning, scheduling, tracking, etc. all at the feature level.

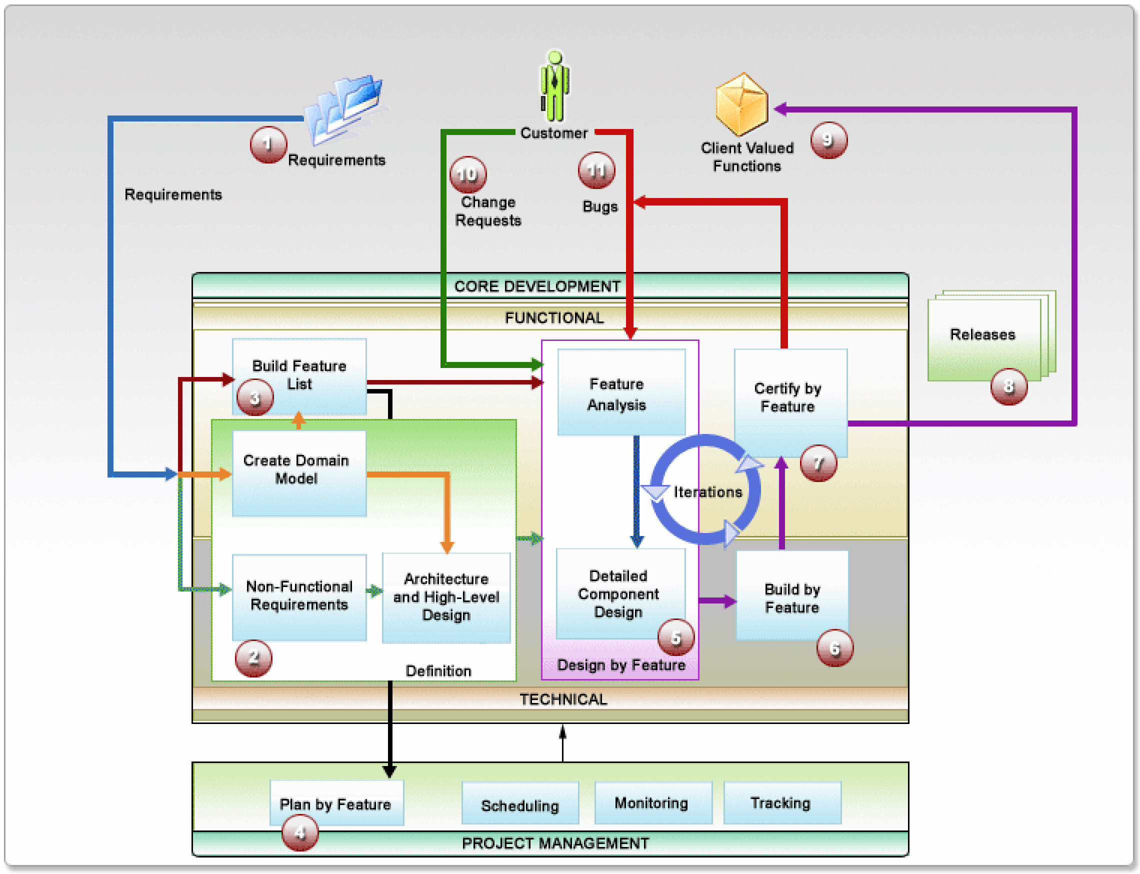


Figure 3: Cognizant FDD process flow

Figure 3 depicts the Cognizant FDD process flow. Each stage of the Cognizant FDD process is shown in detail with clear distinction between the functional and the technical aspects of the process.

1. Requirements Gathering

The process begins with customer requirements gathered like in any other development process. The requirements diverge into the functional and the technical streams and are used as inputs for the next stages in the Cognizant FDD process.

2. Definition

The Definition process is split across the functional and the technical streams. From a functional standpoint, this step involves the creation of a domain model. The domain model is the key input that will help in many of the further stages.

From the technical angle, the requirements are used to define the non-functional requirements (NFRs) of the system. The NFRs list the requirements for performance, scalability, security, etc. that act as the main inputs to the architecture definition process. Using the NFRs and the customer requirements, the architecture and high-level design of the system is defined. At this point, the various frameworks, the logical layers, the deployment tiers, etc. are envisioned and defined in accordance with the requirements.

The high-level design also includes an overall component model. Please note that this is not the elaboration of the components up to the last degree of detail. This may merely be a sketchy interface detail without any further elaboration providing clear assignment of responsibilities across the components. This step also ensures that these components are well-defined and owned by specialized class owners.

3. Build Feature List

The customer requirements also play an important role in the creation of the feature list. The feature list is the main driver for the FDD process because each feature runs through the entire process. The feature list is arrived at in the functional stream using inputs from the domain model. The feature list also provides inputs to the domain model for refining it to align it with the required features.

Feature lists are structured into feature-sets and features. The feature-sets are logical grouping of related features. Each feature by itself runs through the entire process in a single iteration. There could be parallel iterations for developing many features at the same time. The feature lists are composed based on the dependencies between the features and on the basis of customer-defined business priorities.

4. Plan by Feature

The feature lists provide input to the project management stream to define the project development schedules. Based on the grouping defined by the feature-sets and the dependencies between features, the project management team creates the development schedule by planning the number of resources, activities, timelines, deadlines, milestones, etc. The team also sets up processes for monitoring and tracking the development process.

This is similar to the project management in any process, except that it is driven by the feature list.

5. **Design by Feature**

This step is the beginning of the iterative process that each individual feature is run through. This step involves the analysis of the feature by Feature Owners to understand the requirements and expand the feature to the lowest level of detail. The analysis involves a walkthrough of the domain model by the domain expert. This helps the Feature Owners understand the functionality and how it fits into the overall domain model.

After this is done, the Feature Owners along with the class owners arrive at the detailed component design for each feature. This step involves using the understanding of the functionality to the last detail and translating it into a detailed component design including definition of the interfaces, method signatures, implementation details, etc.

Corresponding to the domain model, the actual component where this feature needs to be implemented alone is expanded. Other components untouched by this feature continue to exist with the initial level of details and may be implemented as dummy stubs for construction purposes. This is very critical so that the Feature Owners do not overlap the design for each feature because: a) they might not be the owners of the other feature and hence not understand it correctly, and b) the philosophy is defeated if each feature is not developed individually.

6. **Build by Feature**

The Developers use the detailed component design created in the previous step to construct the code that implements the feature. The Developers will also unit test their code to ensure that the implementation meets the expected functionality. Dummy stubs generate the inputs that might be required from other features for the unit testing.

7. Certify by Feature

Quality Assurance (QA) processes, such as ensuring that all functional and non-functional requirements are met and all standard guidelines followed, etc. are followed as part of the certification step. The certification step might also include other features that have already been developed to perform integration testing with the current feature. Bugs or defects found in the QA process are sent back to the development team for a fix and re-release of the feature to testing.

8. Releases

Once certified by QA, the feature has completed its iteration and is ready for release. Multiple features that are completed in parallel might be grouped together in one release and multiple such releases might be done over the course of the development cycle. This is defined by the project plan and is based on the customer requirements.

9. Client-Valued Functions

The feature releases are ready to be delivered to the clients, as the feature is not just a prototype, but also a full implementation. Imagine achieving this when there are many parts of the system still not defined completely – this is one of the core advantages of FDD.

10. Change Requests

One of the salient features of the Cognizant FDD process is the accommodation of support functions that will be required in enterprise development. One of these is the provision for change requests. The customer might raise change requests during the development phase or after the delivery of a feature. This might have been brought around by some unforeseen change in business requirements. Such change requests can be accommodated in the Cognizant FDD process. Any change request to a feature brings the feature back to the Feature Analysis stage from where the feature is analyzed again with the changed requirements and run through the entire design, coding and testing iteration again.

11. Bug Fix

Bugs could also arise in delivered features. Bugs are run in a similar process to the change requests except that the business requirements remain constant and there is only an analysis of the expected functionality to point out why the feature is not behaving as expected and to fix it through a correction in the design or the code.

Almost every non-agile methodology also tries to achieve the aforementioned. However, the difference is that such a methodology does this implementation for a higher level of granularity (or a group of functionalities together). The feature is the most atomic level at which something can be delivered to the client and hence FDD treats requirements at the lower-most granularity. Also, in the other methodologies, the detailed definition of components for the entire system (or a large chunk of the system – as per the iteration plan) is done completely before it goes to the implementation stage. In FDD, the design is also at the feature level. That makes the whole end-to-end process achievable within a few weeks. Thus, when one feature is being discussed and debated for functional completeness, another may be going through construction or testing. There is no need for a high-level component specification to emerge with all the requirements detailed out prior to moving to the implementation stage. This important difference makes this process agile and capable of accepting change at any stage, because each feature threads can be implemented without any dependency on what stage the other threads are at.

This also means that the same class or component will be modified for different features at the same time. It can lead to chaos and FDD solves this issue by class ownership. It ensures that all the changes to that component at the design level or the implementation level is handled by the same set of people. So there is clear ownership and hence minimal conflict.

One of the significant features of Cognizant FDD is the equal importance given to the functional side. Most of the methodologies bring the functional side at the beginning of the lifecycle and towards the end during the testing and the acceptance phases. Somewhere in between, the functionality is transferred to the Designer and the Developers. Cognizant believes this is the fundamental problem in every methodology and proposes different roles that own the feature throughout its implementation. Due to this difference, other roles have been added. A small comparison between traditional FDD roles and Cognizant FDD roles (wherever there is a difference) and related activities and deliverables is as follows.

Cognizant FDD	Activity/Deliverables	Traditional FDD
Architect	Architecture	Chief Architect
	Design reviews	Chief Programmer
Domain Expert	Domain model	Domain Expert
	Domain walkthrough	Domain Expert
	Feature analysis review	Domain Expert
Developer	Code	Class owner
	Unit test scripts	Class owner
	Installation package	Class owner
Tester	Test cases	Tester
	Test scripts	Tester
	Test results	Tester
Feature Owner	Detailed feature analysis	Chief Programmer
	Design reviews	Chief Programmer
	Feature review	Chief Programmer
	Impacted feature list	Chief Programmer
Designer	High-level design models	Class owner
	Detailed design models	Class owner
	Component specifications	Class owner
	Code review	Class owner
	Design modifications	Class owner
Project Manager	Prioritized feature list	Project Manager
	Component ownership matrix	Project Manager
	Feature-level plans	Project Manager
	Progress report	Project Manager
	Defect report	Project Manager

These additional roles are also aligned to the typical team structure that exists in a business application development team.

Cognizant FDD and MSF for Agile Software Development

MSF process framework is a flexible set of process steps to plan and build business solutions. MSF also advocates iterative development and supports architecture-centric, component-based development. MSF advocates design at 3 levels: conceptual, logical and physical. It also advocates architecture as the center around which the whole system is developed.

One of the critical features of any methodology is the threads of execution. The question that comes up is -- what is the lowermost granularity of execution across different iterations defined in the process? As we have seen earlier, in the case of FDD, it is the feature. Similarly, in RUP, it is use case, and in MSF, it is scenario. However, in MSF use cases or features can also easily integrate. It is this openness in MSF that makes it the ideal platform for building Cognizant FDD.

The upcoming release of Visual Studio Team System (VSTS 2005) that supports the automation of MSF model is an extensible framework that can be used for implementing any custom development process. Cognizant FDD was automated based on the principles and the object model of MSF. Cognizant FDD is more aligned towards feature-driven process as compared to MSF for Agile Software Development, which is a scenario-driven process. We can consider MSF for Agile Software Development as a sample process that captures the essence of MSF principles. However, it is important to understand the meta-model of MSF that supports almost every other process. MSF for Agile Software Development is just one instance of that and Cognizant FDD is another. Any customized process can be built over the same framework.

Since MSF has a meta-model that supports almost all types of processes and it has been automated by VSTS, it becomes the ideal platform to build a custom process such as Cognizant FDD.

On a slightly different note, none of the processes can be an exact copy of a standard methodology. A development process followed within a software development team will be based on the unique process and methodology blueprints followed in that company, characteristics of the end user organization for whom the business application is being developed, their involvement in the development process, skills of the team members, availability of tools, and so on. What is required for the automation of a process is a flexible framework where the good principles from several methodologies can be combined to form a customized process addressing the unique situation at hand. Cognizant has found MSF and VSTS offer such a flexible framework. And Cognizant FDD templates within VSTS are examples of such a customized implementation.

Implementation of Cognizant FDD Using VSTS

A new process methodology within VSTS is defined using a set of externalized XML files that guide the progress of the project throughout its lifecycle. Defining a new methodology means creating or extending various VSTS elements such as project structure, work items, source control settings, groups and permissions, reports, process guidance help, etc.

Overall architecture of the VSTS and the areas where customizations have been done for implementing Cognizant FDD are shown in the following diagram.

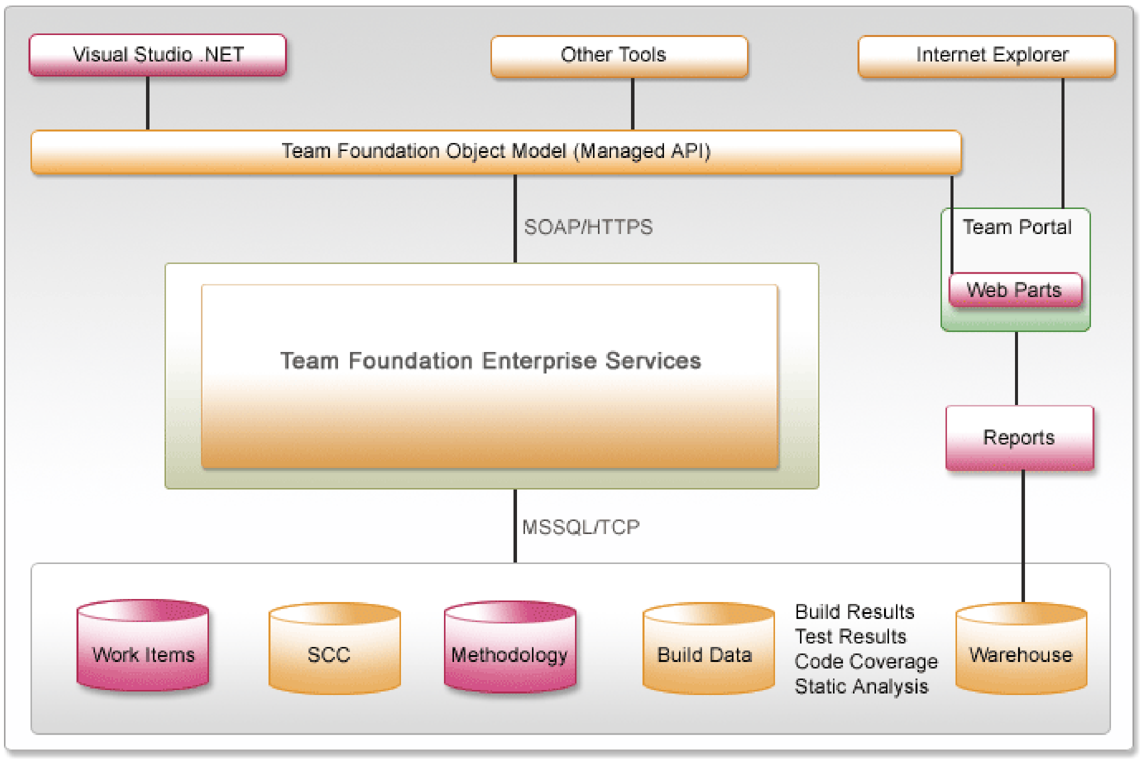


Figure 4: Implementation of Cognizant FDD Using VSTS

The following sections describe the areas of the VSTS that were customized or extended.

Methodology database: This is a repository of all process templates. The methodology templates required for implementing Cognizant FDD were added to this. These templates define all the necessary tasks for a new team project such as work item type creations, portal site creation, reporting site creation, source control setup, etc.

Work items database: This database holds the work items created for a project using Cognizant FDD. The attributes of these work items, such as the state of completion, target end date, etc., are captured in this database for tracking and reporting.

When a new project is started, the initial startup tasks needed for a Cognizant FDD-based project is created by default within this database. Based on the subsequent steps in the process, further work items and tasks are added to the database.

Reports: Every team project has a reporting site created in the Reporting Services server. This site holds all reports defined in the methodology. As part of the customization for Cognizant FDD, several reports have been provided out of the box. These reports can help in tracking and managing the progress of feature implementation at a fine-grained level. Common reports, such as feature status, plan vs. actual, pending tasks, etc., have been provided out of the box.

Team portal: Every team project has a team portal site created using SharePoint Services. This site holds all the process guidance defined in the methodology. Customized process guidance for Cognizant FDD is added to the team portal. This also contains various templates used for documenting the artifacts, such as architecture, domain modeling, etc.

Visual Studio: The Team Explorer augments the user interface of Visual Studio. It is driven by the customizations of work item templates done for Cognizant FDD. The fields displayed the default values, the states of work items, workflows, etc. within Team Explorer are defined by the methodology templates. There is a close tie up between Visual Studio and Team Explorer. For example, while checking in from Visual Studio, all the check-in policies defined in VSTS are applied. If there are violations, then customized messages are displayed.

Cognizant FDD contains its own set of work items, security settings, process guidance help, FDD process reports, source check-in policies, work products templates, project portal, etc. to support the methodology. The following are some of the important changes that were made to define Cognizant FDD.

All work items related to Cognizant FDD, such as feature, change, bug, etc., have been added.

There are several customized queries used by the project team. These include pending tasks, active bugs, etc. Typical Cognizant FDD implementation needs several such queries that are essential for feature-based development. These additional queries are defined based on the work item types.

The initial sets of work items that a Project Manager would need when a project is started are also provided.

A sample set of work items for the entire project lifecycle is also given. This has a sample instance of every task and can act as a guide to the project team.

The roles necessary for Cognizant FDD are defined as groups in VSTS. For each of these groups, appropriate permissions are given. When a new project is created, groups with those permissions are automatically created for that project.

When a new project based on Cognizant FDD is created, a new portal site using SharePoint Portal Services is created automatically. The setting up of a default portal is provided by VSTS and it has been customized to create the portal to be in line with Cognizant FDD-based project document templates and processes. In a similar way, the custom reports are also added to the reporting site. The source control settings that were defined in the process templates are also applied to the source control repository.

Note that all these are one-time tasks and any project that uses this methodology can directly start the development process in a fully automated fashion. When a new project instantiates Cognizant FDD process within VSTS, the following are created by default:

? Complete set of initial tasks
? Complete work items types and behavior (states and transitions)
? Groups and permissions
? Queries

- Reporting site for the project with several reports to help track and manage the project
- Team portal site for the project that contains
    - Document templates
    - Process guidance
    - Web parts for the custom reports
- Source control repository

Further customization can happen at a project level. This could include additional reports for a specific need, adding additional information in the portal site, etc.

Cognizant FDD - Details of the Process

This section explains the implementation of Cognizant FDD using VSTS.

Overview

Microsoft Visual Studio Team System (VSTS) provides tools to support the entire software development process, which includes process guidance, architectures, and life-cycle tools for deploying solutions. VSTS has a seamless integration between the tools and processes and provides a unified e-development experience that also adheres to the process.

Cognizant FDD implemented using VSTS contains customized work items, security settings, process guidance, FDD process reports, source check-in policies, related artifacts, project portal, etc. to support the feature-based development processes.

To understand Cognizant FDD implementation using VSTS, it is important to understand the underlying meta-model. The overall process can be divided into 3 distinct parts.

Roles: Roles represent the actors executing the process steps. Cognizant FDD has distinct roles that perform various tasks based on their unique skills. A project team comprises various members playing these different roles. It is possible to have an individual team member playing different roles.

Work Items: These are threads of execution that are used to manage and track the project over its progress through various phases. At the lowermost level, it can be a task. However, at a higher level, it can be any thread partitioning the software development. As we have seen in the earlier discussions, the threads can be use cases (RUP), features (FDD) or scenarios (MSF for Agile Software Development). Thus, work items are collection of tasks (such as definition or startup) or can be a thread of execution for a support process (such as bug or change).

Understanding the concept of work item is very important. The states of completion of a work item are managed and tracked during the execution of the project. So we can think of a work item as a chunk of tasks that need to be executed to achieve the end result. The status of a software development project can be considered as the combined status of the work items. What we gain from the process automation frameworks such as VSTS is the ability to track the various attributes and states of the work items as it moves from role to role performing certain tasks.

Work streams: Work streams are groups of activities connected in some logical manner (such as the design by feature). Many roles participate in the activities in a work stream. Each of these activities can consume and/or produce or change work products. A domain model or architecture consists of sample work products produced during the definition work stream.

The next few sections talk about Cognizant FDD roles, work items and work streams as defined in the VSTS.

Roles

The different roles in Cognizant FDD are as follows.

Architect

The main responsibility of the Architect is to define the overall architecture of the system based on the requirements and matching stakeholder viewpoints. This includes gathering the non-functional requirements for the application, creation of architecture views from multiple views, high-level component model, selection of frameworks, products and tools, etc. The Architect will also be involved in reviewing the design created by the Designers and resolve those issues on design that the Designers cannot resolve themselves.

Domain Expert

The main goal of Domain Experts is to impart their deep knowledge of business to the team at various levels of abstraction depending upon the stage and the role involved. It may involve domain modeling or explaining a complex business rule to a Developer or a Tester. The domain experts will conduct domain walkthroughs during the *design-by-feature* phase and will also be involved in reviewing *feature analysis* done by Feature Owners.

Feature Owner

The main goals of Feature Owners include translation of requirements to features, preparation of detailed requirements and doing a thorough feature analysis. They are purely functional experts interacting with the Designers and Developers to implement a feature. They understand the feature functionality requirement at a granular level and represent the end user regarding feature requirements and completeness. They focus on the business aspect leaving the Developers to focus on technology.

Designer

The main goals of Designers include preparation of high level design models for all services, components and frameworks and creation of detailed design models for features. They are experts in translating the architecture principles to the design of the components. They ensure that multiple and often conflicting functional and non-functional requirements on different components are resolved by applying proven patterns. They own the design for a given feature and actively participate in reviewing the source code developed by Developers. In smaller teams, this role may be combined with that of the Developer.

Developer

The main goal of Developers is to implement the application. Their primary input is the design created by the Designer. Based on the design, the Developers code and unit test the features.

Tester

The main goal of Testers is to independently verify system functions and ensure that they meet the end user requirements. They would be involved in creating test cases and test scripts for a feature. After the implementation of the feature is complete, they run their scripts and certify the feature. If any issues are found, they communicate it back to the concerned members in the team.

Project Manager

The responsibilities of a Project Manager include planning, scheduling, monitoring the progress and reporting status. A Project Manager's primary goal is to create and maintain an environment where the team works productively. The Project Manager prioritizes and sequences the features and recognizes Feature Owners and component owners and assigns them their responsibility. The Project Manager ties the functional and the technical teams together by way of various processes, guidance, communications, and ensures project delivery within the constraints of schedule, effort and cost.

Work Items

Various work items of Cognizant FDD and the state transitions associated with these work items as defined in VSTS are as follows.

Feature

A feature is a type of work item that represents a small client-valued function that can be tracked throughout its life cycle. The Project Manager creates a feature work item during the "plan-by-feature" phase and assigns it to the Feature Owner. The Feature Owner keeps track of the feature/s using this work item. Various design, development and testing tasks are closed by the assignees, after they have linked their work products (enforced by check-in policies) to those tasks. This provides for traceability between the features and their work products.

All the states with transitions for the main flow in the feature work item are as follows.

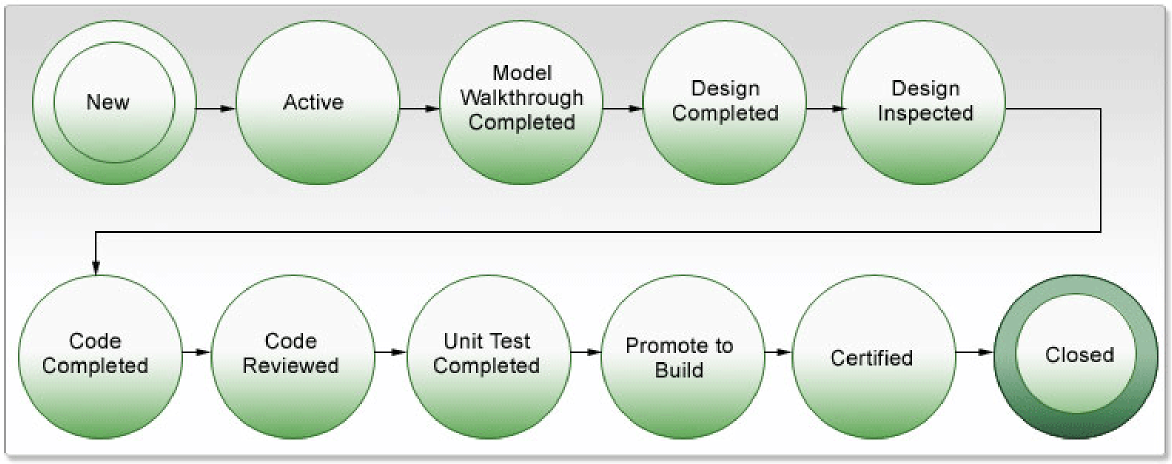


Figure 5: States in Feature Work Item

Descriptions of each of these states are as follows.

State	Description
New	A feature work item will be in a new state when it is created.

Active	When a new feature work item is created using Team Explorer, the status, is automatically set to 'active'.
Model Walkthrough Completed	A feature has this state when the domain model and the high-level component walkthrough to Designers and Feature Owners are complete.
Design Completed	A feature has this state when the Designers have completed the design for that feature.
Design Inspected	A feature has this state when the Architect and the Feature Owner have reviewed the design.
Code Completed	A feature has this state when the Developer has implemented the feature using the design artifacts created by the Designer.
Code Reviewed	A feature has this state when the Designer has reviewed the code and the Developer has implemented those review comments.
Unit Test Completed	A feature has this state when the Developer has performed unit test.
Promote to Build	A feature has this state when the code has been checked into the version control system to go into the build.
Certified	A feature has this state after the Testers have run their test scripts on the promoted build.
Closed	A feature has this state when all activities related to the feature have been completed.

Details of the transition between the states and the roles involved in this transition are as follows.

From State	To State	Transition When	Roles Involved
New	Active	When a new feature work item is created using Team Explorer, the status is automatically set to active.	Project Manager
Active	Model Walkthrough Completed	A model walkthrough is complete.	Project Manager
Model Walkthrough Completed	Design Completed	Design for the feature is done.	Project Manager, Designer
Design	Design	The Architect completes the	Project Manager,

Completed	Inspected	review and the Feature Owner ensures it covers the feature.	Architect, Feature Owner
Design Inspected	Code Completed	The Developer completes the coding.	Project Manager, Developer
Code Completed	Code Reviewed	The Designer reviews the code and the Developer implements those review comments.	Project Manager, Designer, Developer
Code Reviewed	Unit Test Completed	The Developer completes unit testing.	Project Manager, Developer
Unit Test Completed	Promote to Build	A feature is ready for testing.	Project Manager, Feature Owner
Promote to Build	Certified	The Tester has tested the feature successfully.	Project Manager, Tester
Certified	Closed	A feature is completed.	Project Manager, Feature Owner

Note: The transitions depicted here and for the other work items indicate only the main flow. The alternative flows have not been taken into consideration. Users have the flexibility to change to any state at any point of time. For example, when a Developer during coding finds a feature with incorrect design, the Developer can change the state directly to “Model Walkthrough Complete” state, so that the design can change. This way the whole process is reentrant – both MSF and VSTS support this and it is applicable to all work items.

Task

A task means a piece of work that can be assigned to any role. A task work item indicates that a certain work needs to be done. For example, an Architect is assigned a task to create architecture and a Designer is assigned a task to create design artifacts for a feature. On the work item form, certain task types (such as Model Walkthrough, Design, Design Inspection, etc.) are provided and are specific to a few roles. When the task is completed, the assignee checks in the work product related to the task. The check-in policy enforces the check-in to be linked to a work item, that is, the task for which it was produced. The task is closed automatically after the check-in.

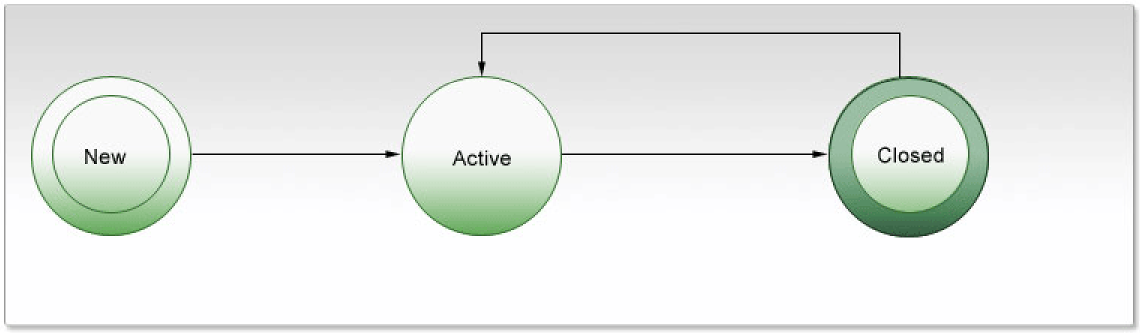


Figure 6: States in the Task Work Item

Descriptions of each of these states are as follows.

State	Description
New	A task will be in the new state when it is created.
Active	When a new task is created using Team Explorer, the status is automatically set to active.
Closed	A task has the closed state when all activities related to it are completed.

Details of transition between the states and the roles involved in the transition are as follows:

From State	To State	Transition When	Roles Involved
New	Active	When a new feature work item is created using Team Explorer, the status is automatically set to active.	Project Manager
Active	Closed	The task is complete or is not required.	All roles
Closed	Active	The task is incomplete or requires additional work.	All roles

Bug

A bug is a work item that communicates a potential problem in the system. The goal of opening a bug is to accurately report bugs in a way that allows the reader to understand the full impact of the problem. The descriptions in the bug report should make it easy to trace through the steps used when the bug was encountered, thus allow it to be easily reproduced. The test results should clearly show the problem has been resolved.

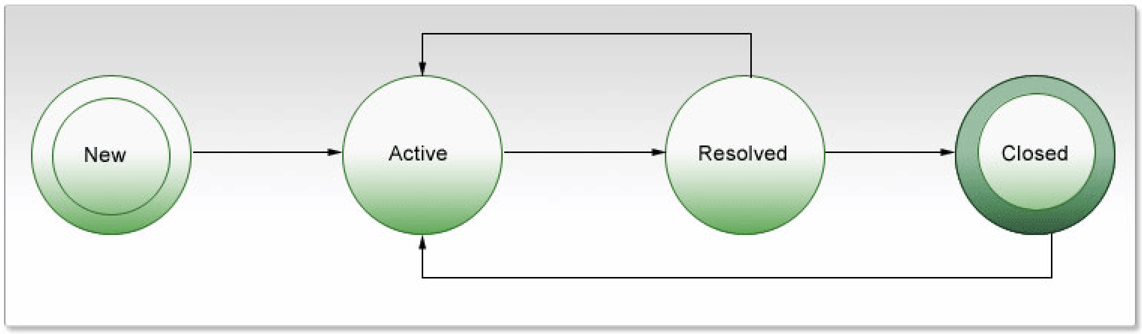


Figure 7: States in the Bug Work Item

Descriptions of each of these states are as follows.

State	Description
New	A task will be in the new state when it is created.
Active	When a new task is created using Team Explorer, the status is automatically set to active.
Resolved	The Developer who has been assigned the bug marks it as Resolved after addressing it.
Closed	A bug is closed if it has been addressed completely.

Details of transition between the states and the roles involved in the transition are as follows:

From State	To State	Transition When	Roles Involved
New	Active	When a new feature work item is created using Team Explorer, the status is automatically set to active.	Tester
Active	Resolved	The bug has been fixed.	Developer
Resolved	Active	The fix is incomplete or incorrect.	Tester
Resolved	Closed	The author of the bug has verified it.	Tester
Closed	Active	The bug has to be tested again or tested for regression.	

Change

Change work item deals with handling change requests initiated by the clients/users. Change requests may require modifications to features completed or currently in progress, which need to be tracked separately. Change requests typically involve changes to the design and the code. Changes can affect the architecture of the system also. But architectural changes, even if they happen, are very rare. Hence only design and code changes are considered for changes.

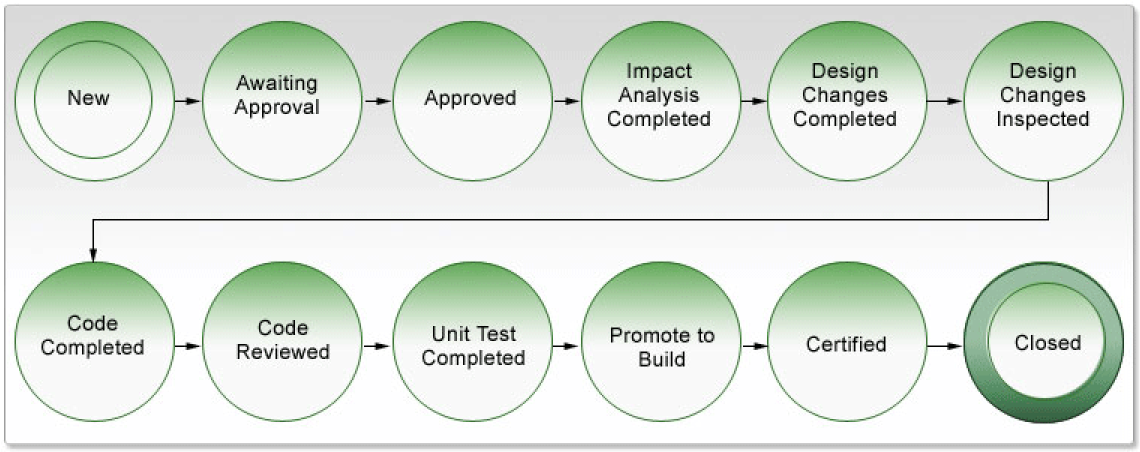


Figure 8: States in the Change Work Item

Descriptions of each of these states are as follows.

State	Description
New	A change request will be in the new state when it is created.
Awaiting approval	When a new change request is saved using Team Explorer, the status is automatically set to awaiting approval.
Approved	A change request approved by the Project Manager goes to the approved state.
Impact Analysis Completed	A change request goes to this state after all architectural and design changes are analyzed and identified.
Design Changes Completed	A change request goes to this state after all design changes for this change are completed.
Design Changes	A change request goes to this state after all design changes for this change are inspected.

Inspected	
Code Completed	A change request goes to this state after all code changes for this change are completed.
Code Reviewed	A change request has code reviewed state when the Designer reviews the code and the Developer implements those review comments.
Unit Test Completed	A change request has this state when the Developer performs unit test.
Promote to Build	A change request has this state when the code is checked into the version control system to go into the build stage.
Certified	A change request has this state after the Testers run their test scripts on the promoted build.
Closed	A change request is closed if it has been addressed completely.

Details of transition between the states and the roles involved in the transition are as follows:

From State	To State	Transition When	Roles Involved
New	Awaiting Approval	When a new change management work item is created using Team Explorer, the status is automatically set to awaiting approval.	Project Manager
Awaiting approval	Approved	The change has been approved.	Project Manager
Approved	Impact Analysis Completed	The impact of the change on the architecture and design has been analyzed.	Project Manager, Architect, Feature Owner, Designer
Impact Analysis Completed	Design Changes Completed	Designer completes the design changes.	Project Manager, Designer
Design Changes Completed	Design Changes Inspected	The architect has reviewed the design changes.	Project Manager, Designer, Architect
Design Changes Inspected	Code Completed	The Developer has completed the construction.	Project Manager, Developer
Code	Code	The Designer has reviewed the	Project Manager,

Completed	Reviewed	code and the Developer has implemented those review comments.	Designer, Developer
Code Reviewed	Unit Test Completed	The Developer has completed unit testing.	Project Manager, Developer
Unit Test Completed	Promote to Build	A change request is ready for testing.	Project Manager, Feature Owner
Promote to Build	Certified	The Tester has tested the changes successfully.	Project Manager, Tester
Certified	Closed	A change request has been completed.	Project Manager, Feature Owner

Work Streams

Work streams are a group of related activities associated with one or more roles. Each of the work streams of Cognizant FDD as implemented in VSTS is explained as follows.

Definition

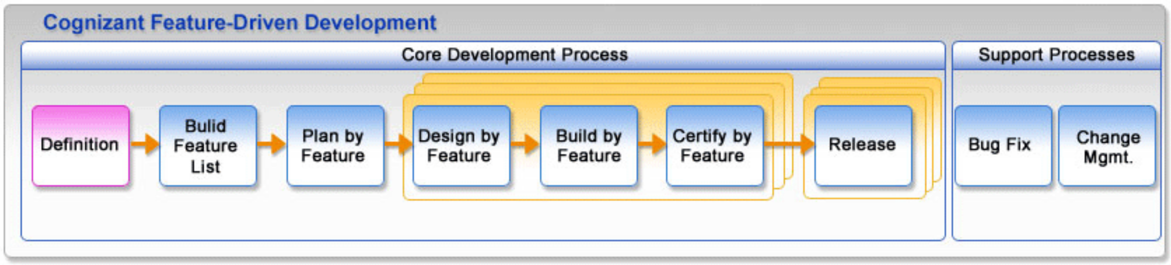


Figure 9: Definition Work Stream

This work stream includes an initial set of project startup activities. It also contains specific tasks necessary to build the basic architecture and domain model foundations necessary for feature-wise iterative development. The initial activities involve the Project Manager to form the teams for domain modeling, feature, design, architecture, development and testing teams.

The roles involved and their activities in this work stream are as follows:

Project Manager:

- Identify team members for performing various roles and set up domain, architecture, design and Feature Owner teams.

- Set up permissions for the team.
- Allocate tasks to domain experts, Feature Owners, architects and Designers.

Domain Expert:

- Build a Domain model based on requirements.
- Conduct a domain walkthrough with the architects.

Architect:

- Create a complete architecture foundation for the project.
- Incorporating the non-functional requirements related changes in the architecture.
- Select framework, products and tools.

Feature Owner:

- Prepare detailed high-level analysis of the requirements.

Designer:

- Prepare high-level design model for the entire project.

The following figure represents activities in the Definition work stream and the roles performing them.

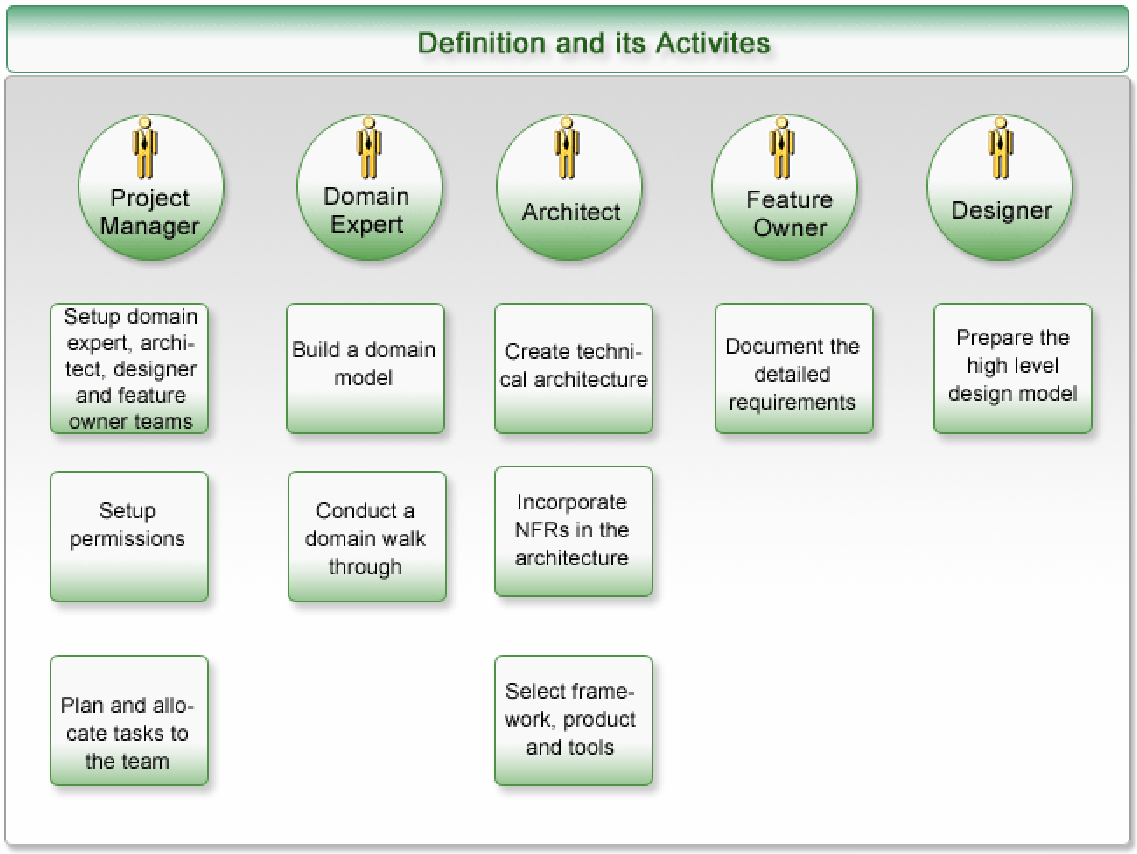
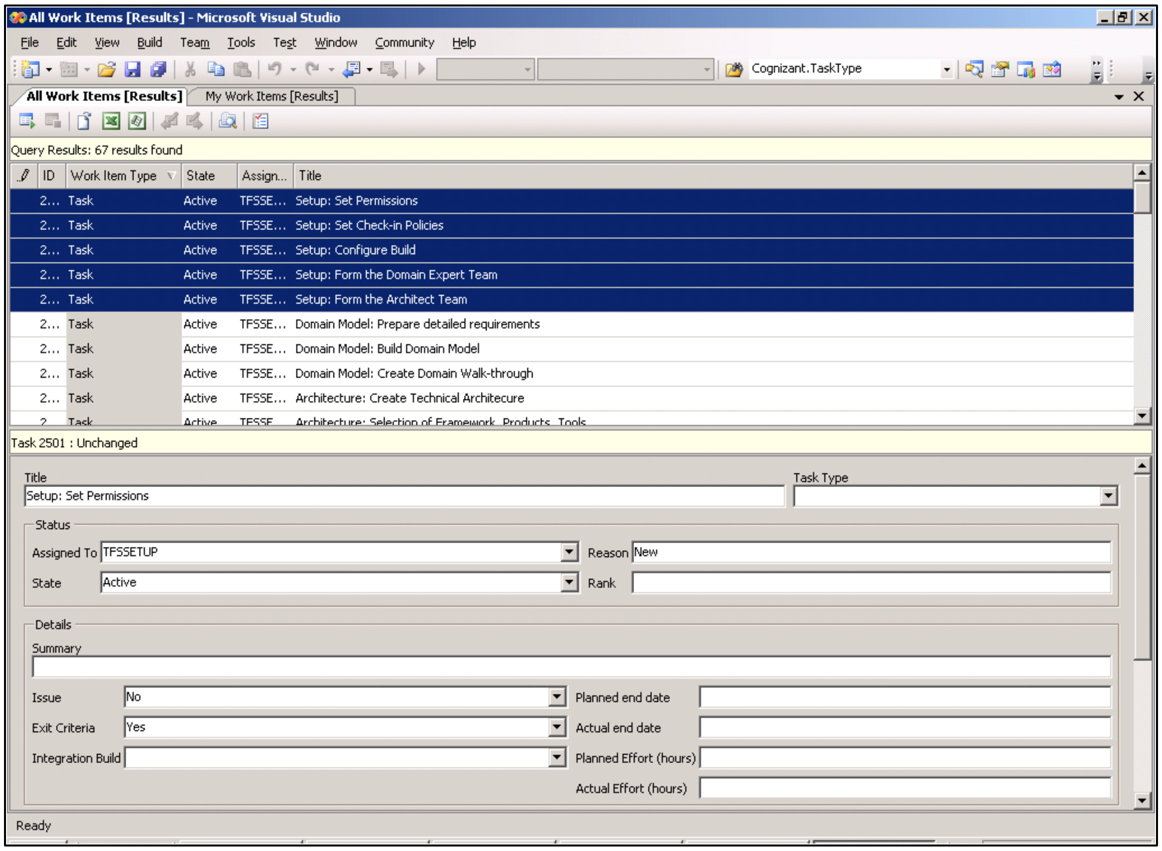

Definition and its Activites
Project Manager
Domain Expert
Architect
Feature Owner
Designer
Setup domain expert, architect, designer and feature owner teams
Build a domain model
Create technical architecture
Document the detailed requirements
Prepare the high level design model
Setup permissions
Conduct a domain walk through
Incorporate NFRs in the architecture
Plan and allocate tasks to the team
Select framework, product and tools

Figure 10: Role-based Activities in Definition Work Stream

The following figure explains how definition tasks are represented in the VSTS. The figure shows the way VSTS displays the initial set of project management tasks created for Cognizant FDD. The beginnings of these task titles convey the intent of those tasks (such as project startup activities, domain model, architecture, etc.).


All Work Items [Results] - Microsoft Visual Studio
File Edit View Build Team Tools Test Window Community Help
Cognizant.TaskType
All Work Items [Results]
My Work Items [Results]

Query Results: 67 results found

ID	Work Item Type	State	Assign...	Title
2...	Task	Active	TFSSE...	Setup: Set Permissions
2...	Task	Active	TFSSE...	Setup: Set Check-in Policies
2...	Task	Active	TFSSE...	Setup: Configure Build
2...	Task	Active	TFSSE...	Setup: Form the Domain Expert Team
2...	Task	Active	TFSSE...	Setup: Form the Architect Team
2...	Task	Active	TFSSE...	Domain Model: Prepare detailed requirements
2...	Task	Active	TFSSE...	Domain Model: Build Domain Model
2...	Task	Active	TFSSE...	Domain Model: Create Domain Walk-through
2...	Task	Active	TFSSE...	Architecture: Create Technical Architecure
2...	Task	Active	TFSSE...	Architecture: Selection of Framework, Products, Tools

Task 2501 : Unchanged

Title: Setup: Set Permissions
Task Type:

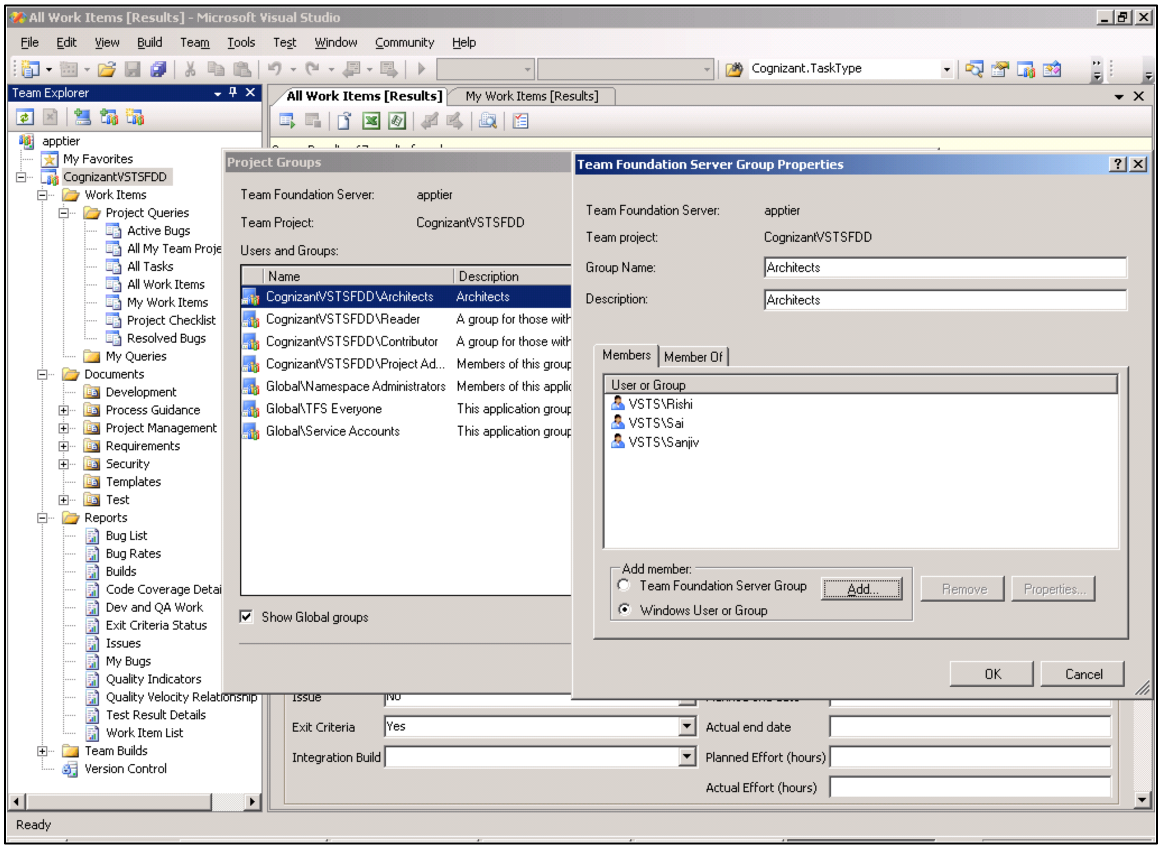
Status
Assigned To: TFSSETUP
Reason: New
State: Active
Rank:

Details
Summary:
Issue: No
Planned end date:
Exit Criteria: Yes
Actual end date:
Integration Build:
Planned Effort (hours):
Actual Effort (hours):

Ready

Figure 11: Initial set of project management tasks

The highlighted items are the initial tasks for the Project Manager, which are required to be completed before initiation of a project. The following image shows how a Project Manager creates roles and assigns permissions to them in VSTS. This is part of the team formation.



All Work Items [Results] - Microsoft Visual Studio

File Edit View Build Team Tools Test Window Community Help

Cognizant.TaskType

Team Explorer

- apptier
  - My Favorites
  - CognizantVSTSFDD
    - Work Items
      - Project Queries
        - Active Bugs
        - All My Team Proje
        - All Tasks
        - All Work Items
        - My Work Items
        - Project Checklist
        - Resolved Bugs
      - My Queries
    - Documents
      - Development
      - Process Guidance
      - Project Management
      - Requirements
      - Security
      - Templates
      - Test
    - Reports
      - Bug List
      - Bug Rates
      - Builds
      - Code Coverage Detai
      - Dev and QA Work
      - Exit Criteria Status
      - Issues
      - My Bugs
      - Quality Indicators
      - Quality Velocity Relationship
      - Test Result Details
      - Work Item List
    - Team Builds
    - Version Control

All Work Items [Results] | My Work Items [Results]

Project Groups

Team Foundation Server: apptier

Team Project: CognizantVSTSFDD

Users and Groups:

Name	Description
CognizantVSTSFDD\Architects	Architects
CognizantVSTSFDD\Reader	A group for those with
CognizantVSTSFDD\Contributor	A group for those with
CognizantVSTSFDD\Project Ad...	Members of this group
Global\Namespace Administrators	Members of this applic
Global\TFS Everyone	This application group
Global\Service Accounts	This application group

☑ Show Global groups

Team Foundation Server Group Properties

Team Foundation Server: apptier

Team project: CognizantVSTSFDD

Group Name: Architects

Description: Architects

Members | Member Of

User or Group
VSTS\Rishi
VSTS\Sai
VSTS\Sanjiv

Add member:
○ Team Foundation Server Group
● Windows User or Group

Add... Remove Properties...

OK Cancel

Issue	No		
Exit Criteria	Yes	Actual end date	
Integration Build		Planned Effort (hours)	
		Actual Effort (hours)	

Ready

Figure 12: Setting Up of Roles and Permissions

Build Feature List

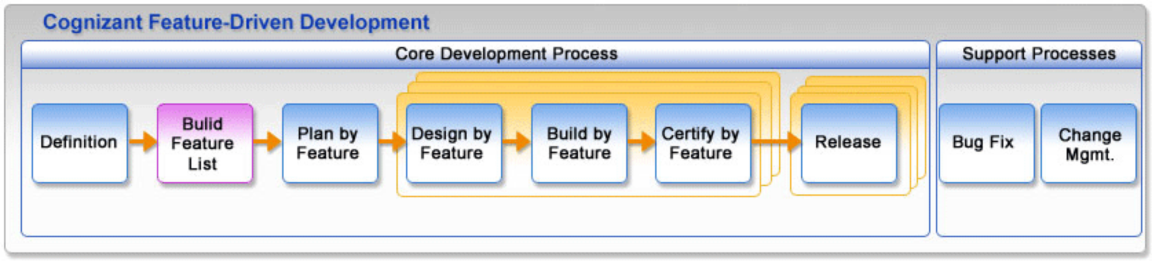


Figure 13: Build Feature List Work Stream

This work stream is an initial project-wide activity to identify all the features needed to support the requirements. The Project Manager and the Feature Owners are actively involved in this work stream and receive inputs from the Domain Expert. Here, the team breaks down the overall requirements into a number of areas (major feature sets). Each area is further broken down into a number of activities (feature sets). Each step within an activity is identified as a feature. The work product is a hierarchically categorized feature list.

The roles involved and their activities in this work stream are:

Feature Owner:

- Translate requirements to features.

Project Manager:

- Define the team structure.
- Prioritize and sequence the features.
- Create a list of Feature Owners and component owners and assign responsibility.

The following figure represents activities in the Build Feature List work stream and the roles performing them.

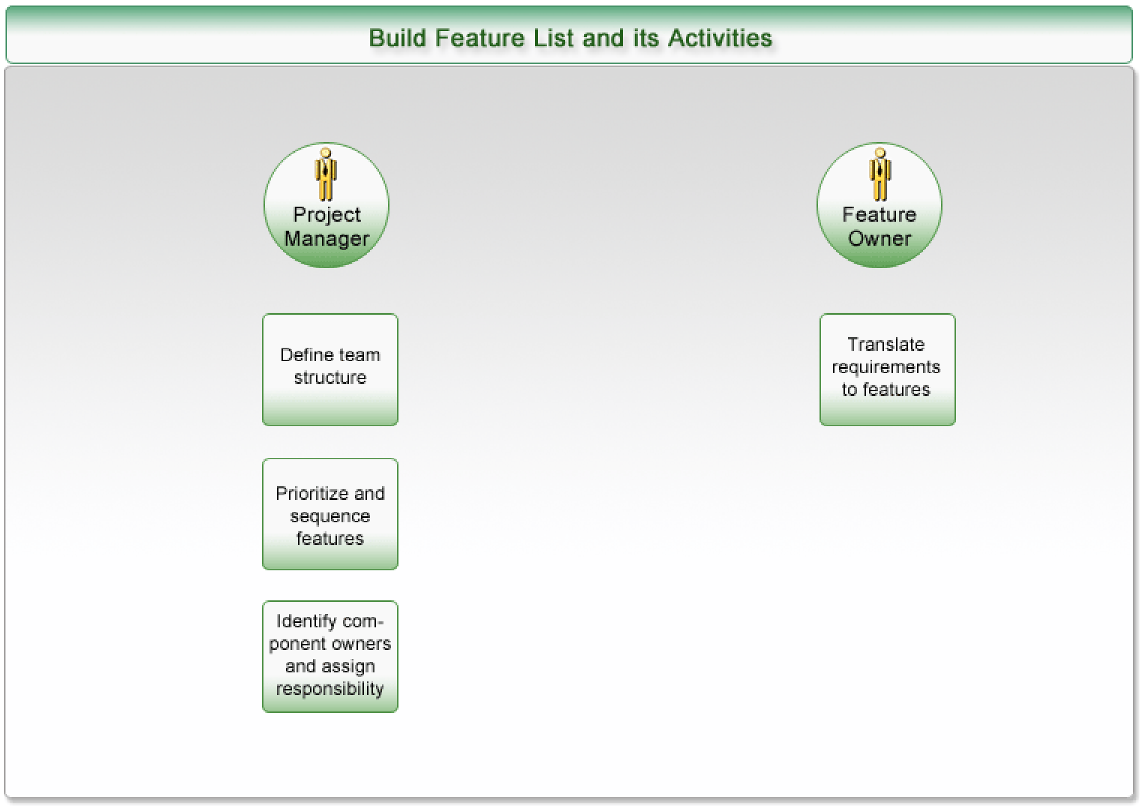

Build Feature List and its Activities
Project Manager
Feature Owner
Define team structure
Prioritize and sequence features
Identify component owners and assign responsibility
Translate requirements to features

Figure 14: Role-based Activities in Build Feature List Work Stream

Plan by Feature

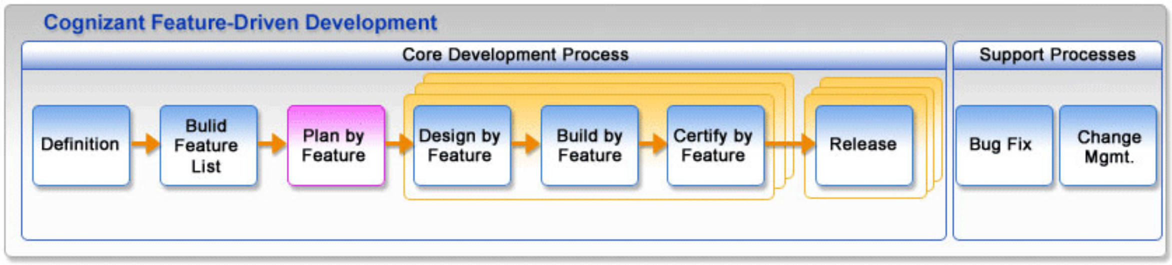


Figure 15: Plan by Feature Work Stream

This work stream is another initial project-wide activity to produce the development plan. The Project Manager and the Feature Owners will be actively involved in this work stream and will receive inputs from the rest of the team.

The roles involved and their activities in this work stream are:

Project Manager:

- Estimate and schedule features.
- Set milestones for "design by feature" and "build by feature" for all features.
- Prepare feature level plan and assign features to Feature Owner.
- Create a list of components along with the Developers who own those. Class ownership is a key feature of FDD.

The following figure represents activities in the Plan by Feature work stream and the roles performing them.

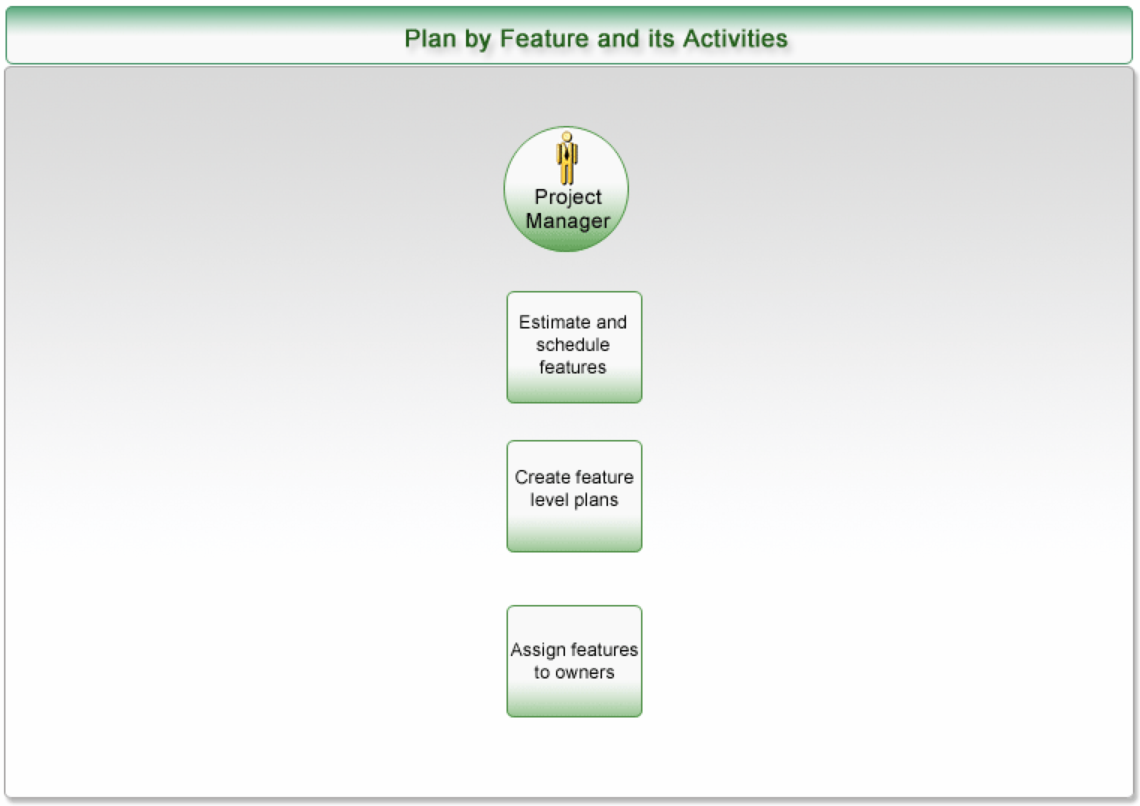
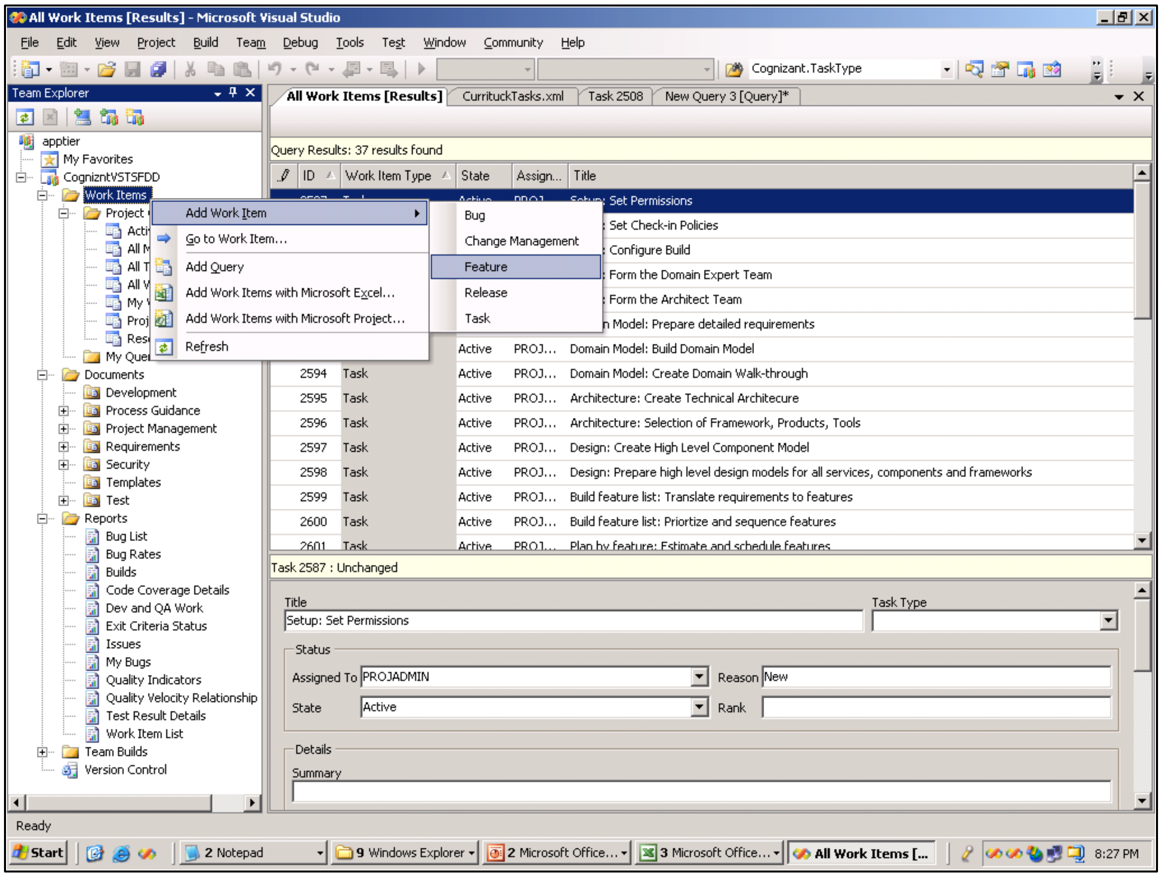

Plan by Feature and its Activities
Project Manager
Estimate and schedule features
Create feature level plans
Assign features to owners

Figure 16: Role-based Activities in Plan by Feature Work Stream

The following figure shows how a new feature will be created as a work item. A Feature Owner can be assigned to a feature and also the status of feature in a work stream can be tracked.



Query Results: 37 results found

ID	Work Item Type	State	Assign...	Title
				Setup: Set Permissions
				Set Check-in Policies
				Configure Build
				Form the Domain Expert Team
				Form the Architect Team
				Model: Prepare detailed requirements
		Active	PROJ...	Domain Model: Build Domain Model
2594	Task	Active	PROJ...	Domain Model: Create Domain Walk-through
2595	Task	Active	PROJ...	Architecture: Create Technical Architecure
2596	Task	Active	PROJ...	Architecture: Selection of Framework, Products, Tools
2597	Task	Active	PROJ...	Design: Create High Level Component Model
2598	Task	Active	PROJ...	Design: Prepare high level design models for all services, components and frameworks
2599	Task	Active	PROJ...	Build feature list: Translate requirements to features
2600	Task	Active	PROJ...	Build feature list: Priortize and sequence features
2601	Task	Active	PROJ...	Plan by feature: Estimate and schedule features

Task 2587 : Unchanged

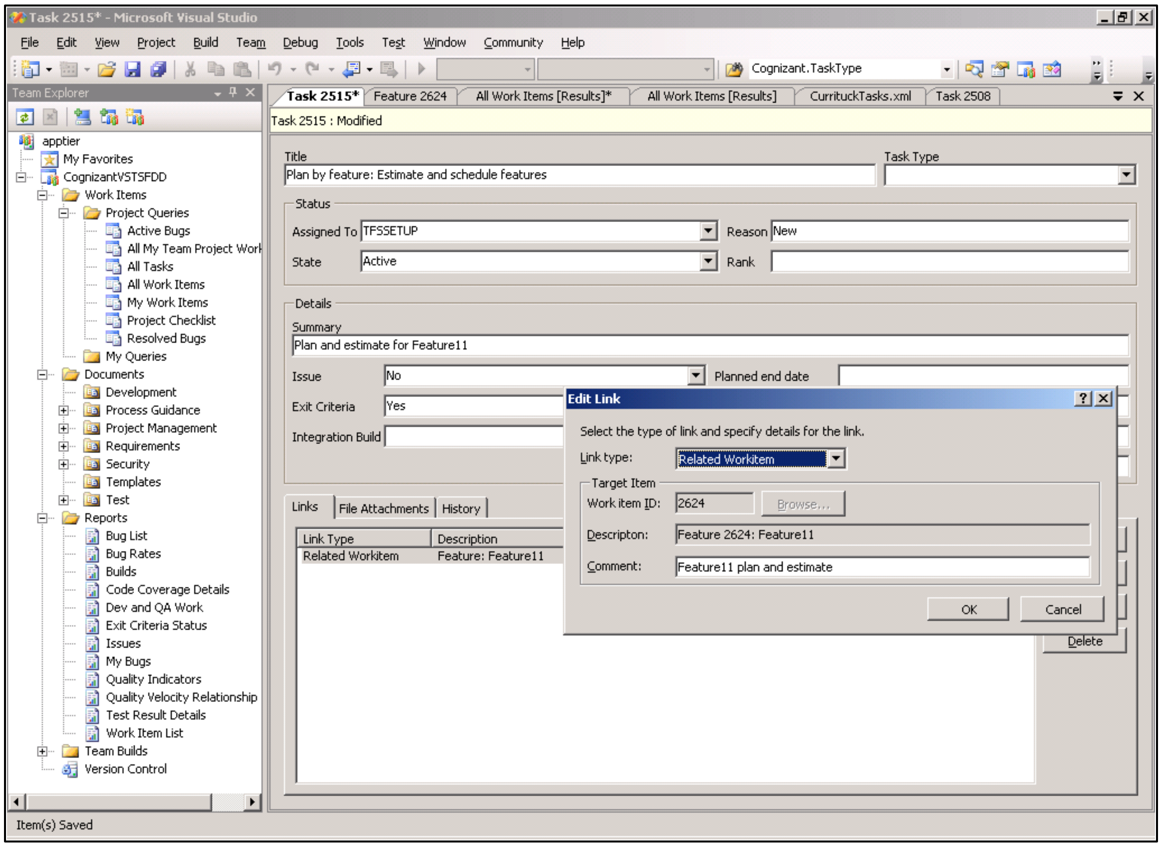
Title: Setup: Set Permissions
Task Type:

Status
Assigned To: PROJADMIN
Reason: New
State: Active
Rank:

Details
Summary:

Figure 17: Creating a New Feature

For each of the features, a plan is created and linked to the task. This task is linked to the actual feature for which the plan and the estimate are being made. This linking is shown in the following figure.



Task 2515 : Modified

Title: Plan by feature: Estimate and schedule features
Task Type:

Status
Assigned To: TFSSETUP
Reason: New
State: Active
Rank:

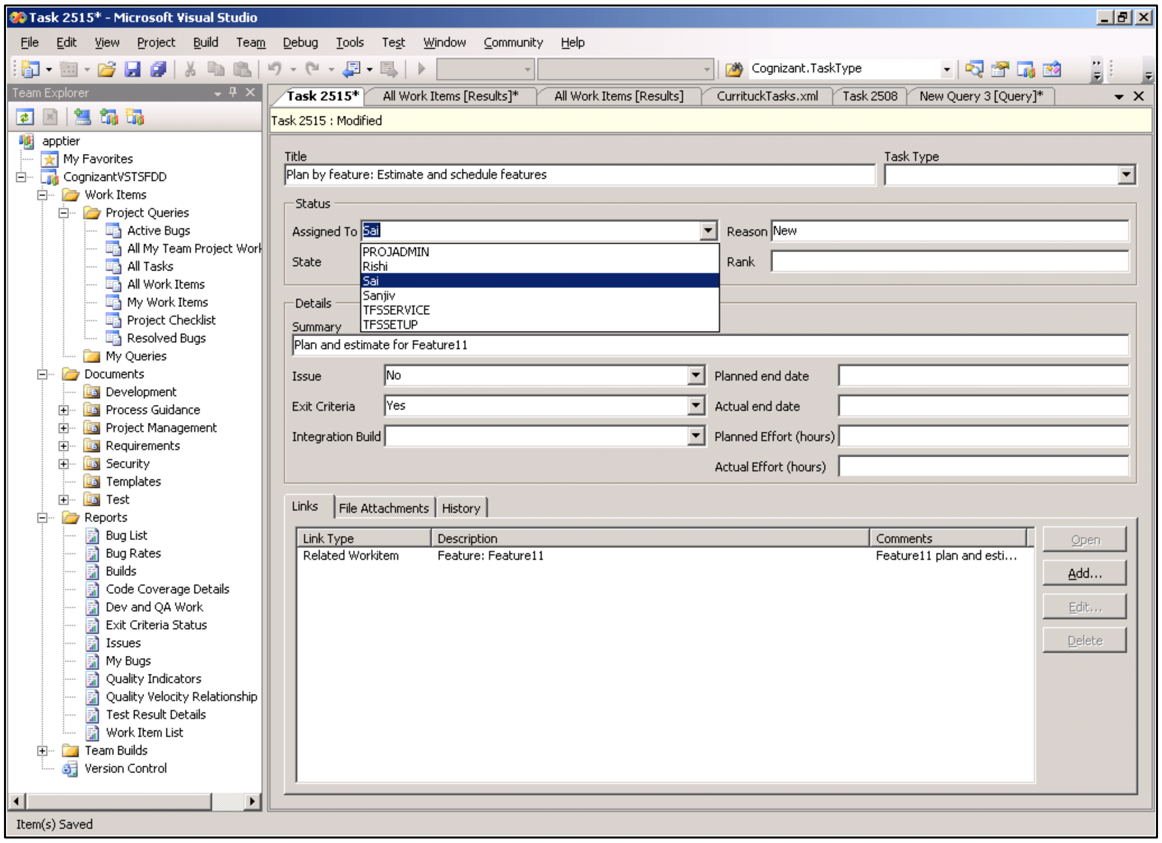
Details
Summary: Plan and estimate for Feature11
Issue: No
Planned end date:
Exit Criteria: Yes
Integration Build:

Link Type	Description
Related Workitem	Feature: Feature11

Select the type of link and specify details for the link.
Link type: Related Workitem
Target Item
Work item ID: 2624
Descripton: Feature 2624: Feature11
Comment: Feature11 plan and estimate

Figure 18: Linking the Feature Plan to the Feature

After the plan is created, the tasks are allocated to the team based on the roles as per the feature plan. This allocation is shown in the following figure.



Task 2515 : Modified

Title	Plan by feature: Estimate and schedule features
Task Type	

Status

Assigned To	Sai	Reason	New
State		Rank	

Assigned To list: PROJADMIN, Rishi, Sai, Sanjiv, TFSSERVICE, TFSSETUP

Details

Summary: Plan and estimate for Feature11

Issue	No	Planned end date	
Exit Criteria	Yes	Actual end date	
Integration Build		Planned Effort (hours)	
		Actual Effort (hours)	

Link Type	Description	Comments
Related Workitem	Feature: Feature11	Feature11 plan and esti...

Figure 19: Assigning Tasks as per the Feature Plan

Design by Feature

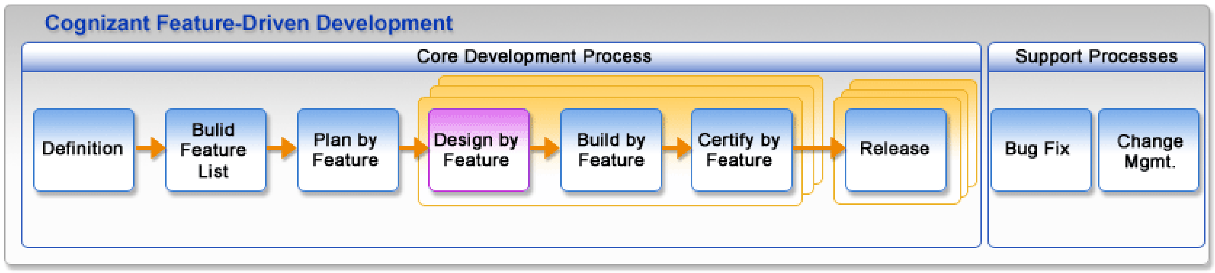


Figure 20: Design by Feature Work Stream

This work stream is a per-feature activity to produce design artifacts required for the feature. The domain expert walks the feature team with an overview of the domain area for the feature in consideration. The Feature Owner takes a feature and does a detailed feature analysis. The Feature Owner identifies the components likely to be involved and contacts the corresponding Designer. The Designer works out to create a detailed design model. The architect and Feature Owner review this design model created by the Designer.

The roles involved and their activities in this work stream are:

Domain expert:

- The domain expert conducts domain walkthrough.

Feature Owner:

- Conduct a detailed feature analysis.
- Coordinate with Designers for realization of design models for the features.
- Review design artifacts created by Designers.

Designer:

- Create detailed design models for the feature.

Architect:

- Review design artifacts created by the Designers.

Project Manager:

- Monitors and tracks the progress of all these activities.

The following figure represents activities in the Design by Feature work stream and the roles performing them.

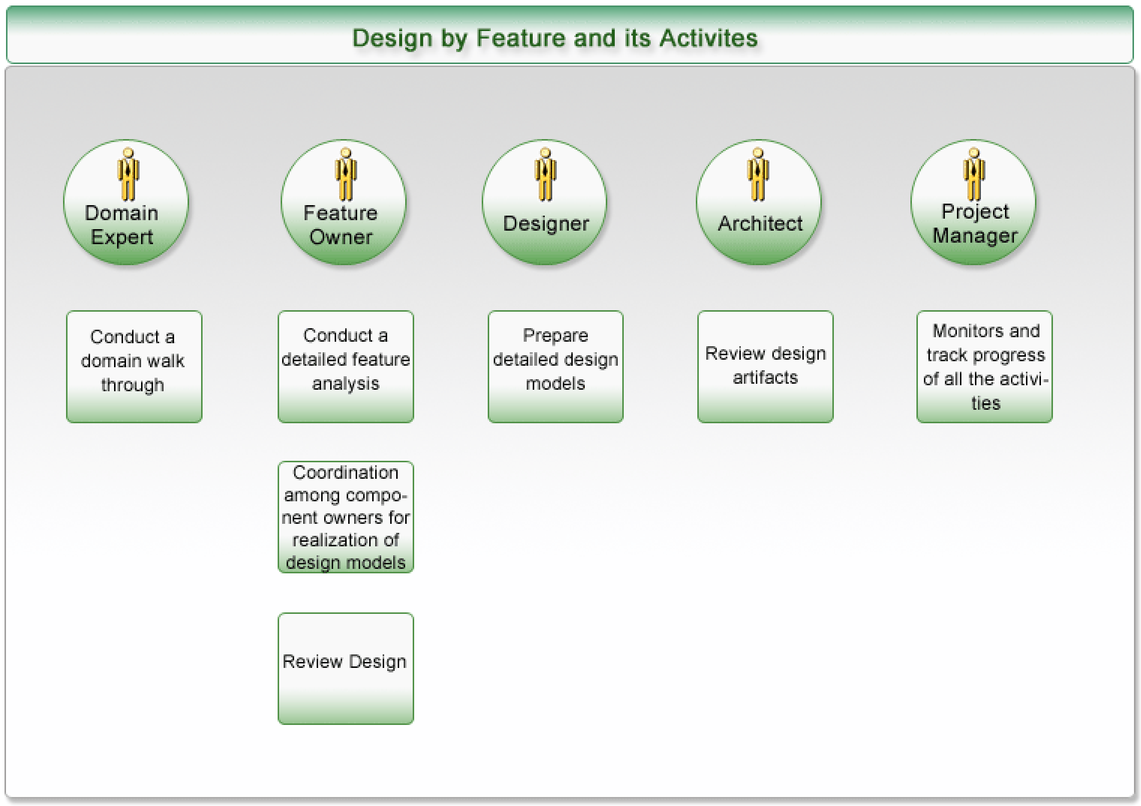

Design by Feature and its Activites
Domain Expert
Feature Owner
Designer
Architect
Project Manager
Conduct a domain walk through
Conduct a detailed feature analysis
Prepare detailed design models
Review design artifacts
Monitors and track progress of all the activi-ties
Coordination among compo-nent owners for realization of design models
Review Design

Figure 21: Role-based Activities in Design by Feature Work Stream

Design model is created using the class designer in Visual Studio for each design task. It is important to note that one of the powerful features of VSTS implementation is tying up of the modeling, coding and testing tools along with the process. So a Designer can create the model and attach or link it to the design task. It is equally important to note that VSTS is quite open in this aspect and is not tied to Visual Studio. It is possible to build the design models in Rational XDE or any other tools and link it to the design task.

The following figure shows the class diagram created within Visual Studio for a design task.

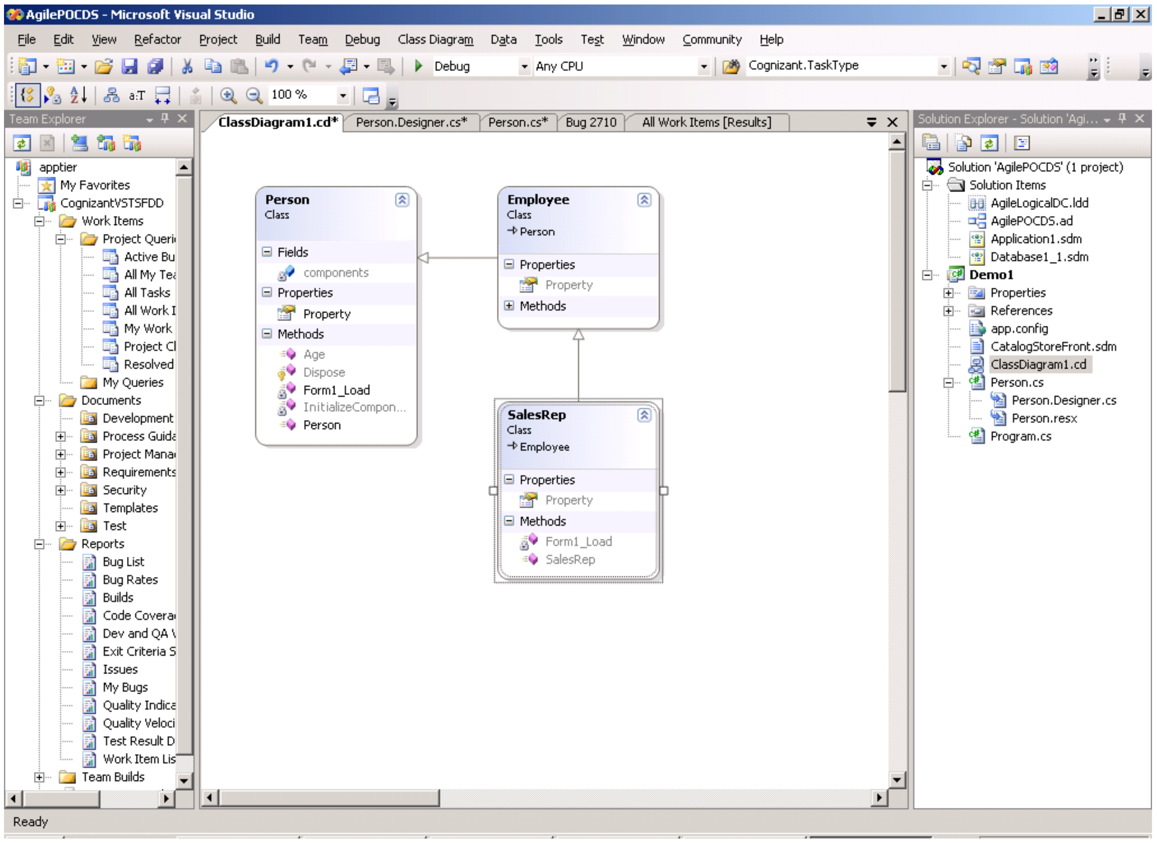
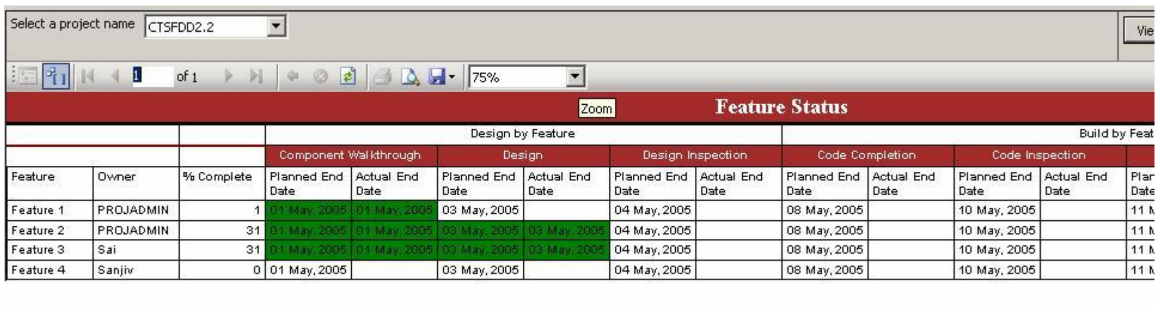


Figure 22: Design for a Feature

From a project management perspective, it is easy to see how various features are progressing across the phases. If there are time slippages, they are shown in red and the Project Manager will be able to take corrective measures. It is this absolute transparency on the progress of the tasks at the feature level that makes FDD a very predictable process.

The following is a snapshot of the custom report from VSTS showing the status of the features as captured during the Design by Feature phase.


Select a project name CTSFDD2.2

1 of 1 75%

Zoom **Feature Status**

			Design by Feature						Build by Feat				
			Component Walkthrough		Design		Design Inspection		Code Completion		Code Inspection		
Feature	Owner	% Complete	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Plar Date
Feature 1	PROJADMIN	1	01 May, 2005	01 May, 2005	03 May, 2005		04 May, 2005		08 May, 2005		10 May, 2005		11 M
Feature 2	PROJADMIN	31	01 May, 2005	01 May, 2005	03 May, 2005	03 May, 2005	04 May, 2005		08 May, 2005		10 May, 2005		11 M
Feature 3	Sai	31	01 May, 2005	01 May, 2005	03 May, 2005	03 May, 2005	04 May, 2005		08 May, 2005		10 May, 2005		11 M
Feature 4	Sanjiv	0	01 May, 2005		03 May, 2005		04 May, 2005		08 May, 2005		10 May, 2005		11 M

Figure 23: Feature Status Report

Build by Feature

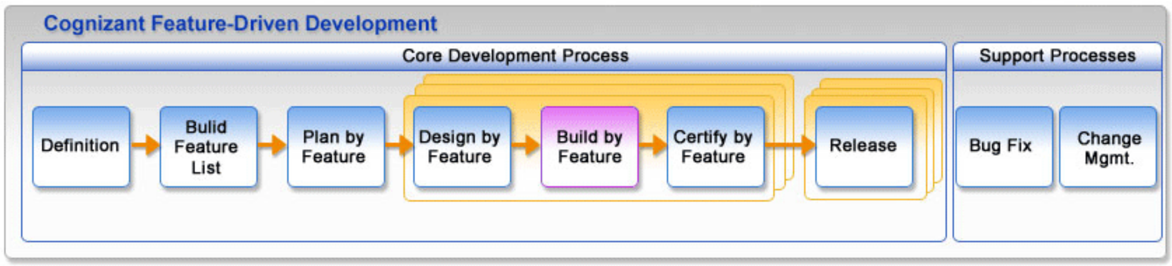


Figure 24: Build by Feature Work Stream

This work stream is a per feature activity to produce a completed client-valued function (feature). Working from the design artifacts created in the "design by feature" work stream, the Developers implement the feature. The code developed is then unit tested by the Developer and inspected by the Designer. After a successful code inspection, the code will be promoted to the build.

The roles involved and their activities in this work stream are:

Developer:

- Implement the feature based on the design artifacts.
- Unit test the feature.

Designer:

- Review the source code developed by the Developers.

Feature Owners:

- Coordinate between component Developers for implementation of the feature.
- Verify feature implementation.
- Promote the feature to build when the code is complete and unit tested.

Project Manager:

- Monitor and track the progress of all these activities.

The following figure represents activities in the Build by Feature work stream and the roles performing them.

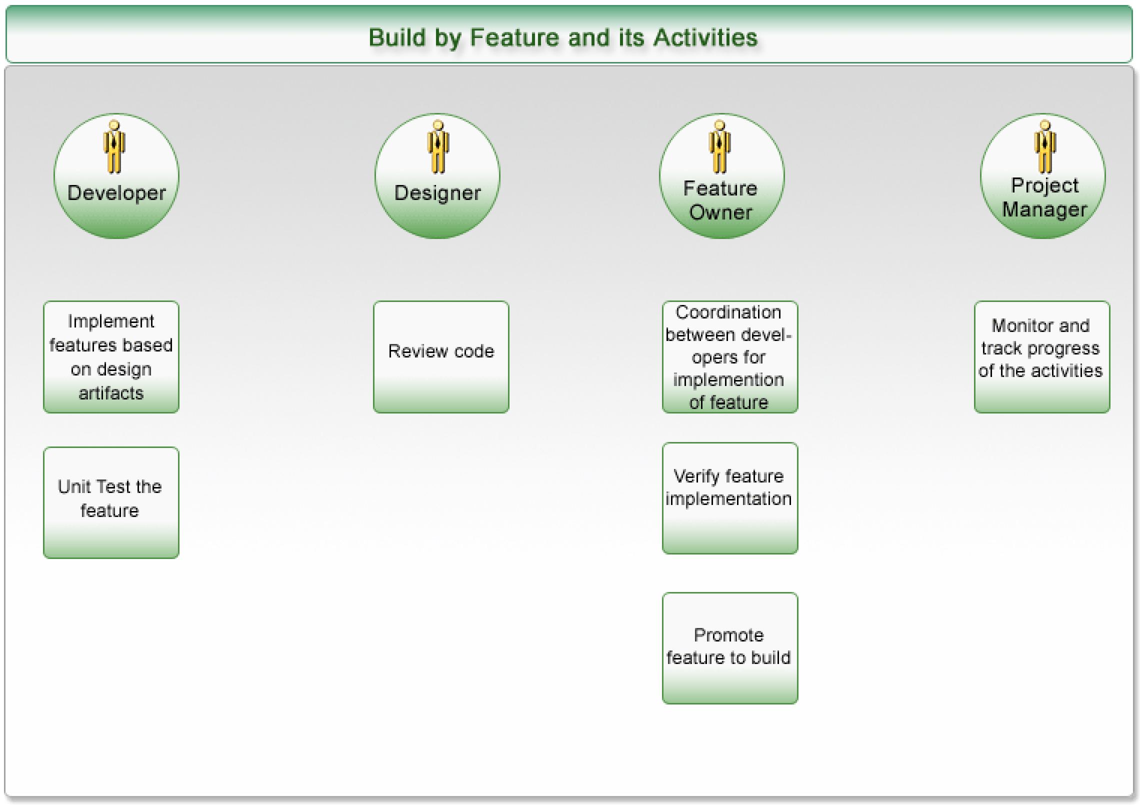
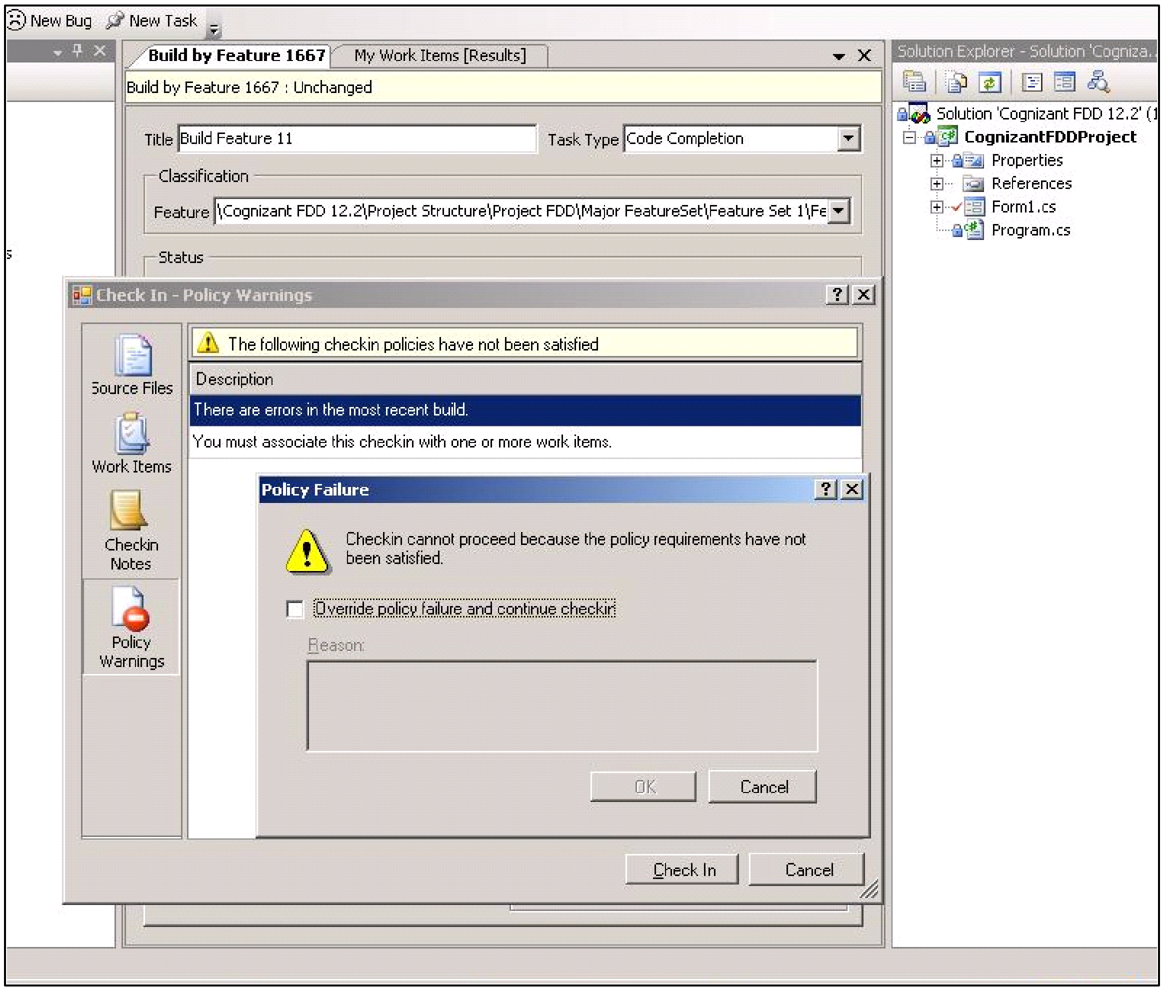

Build by Feature and its Activities
Developer
Designer
Feature Owner
Project Manager
Implement features based on design artifacts
Unit Test the feature
Review code
Coordination between devel-opers for implemention of feature
Verify feature implementation
Promote feature to build
Monitor and track progress of the activities

Figure 25: Role-based Activities in Build by Feature Work Stream

In this phase also, there is a close tie up between tools such as Visual Studio, Source Control, etc. and VSTS that makes the integrated environment seamless and powerful. For example, various policies can be implemented at the check-in level (such as not to allow check-in when there are build errors). Also a Project Manager can create them at any time via the team project settings. Check-in policies can be enforced for all check-ins for the project. Users in the project are provided the following details if they do not comply with it.



Build by Feature 1667 : Unchanged

Title: Build Feature 11 | Task Type: Code Completion

Classification

Feature: \Cognizant FDD 12.2\Project Structure\Project FDD\Major FeatureSet\Feature Set 1\Fe

Status

Check In - Policy Warnings

The following checkin policies have not been satisfied

Description
There are errors in the most recent build.
You must associate this checkin with one or more work items.

Policy Failure

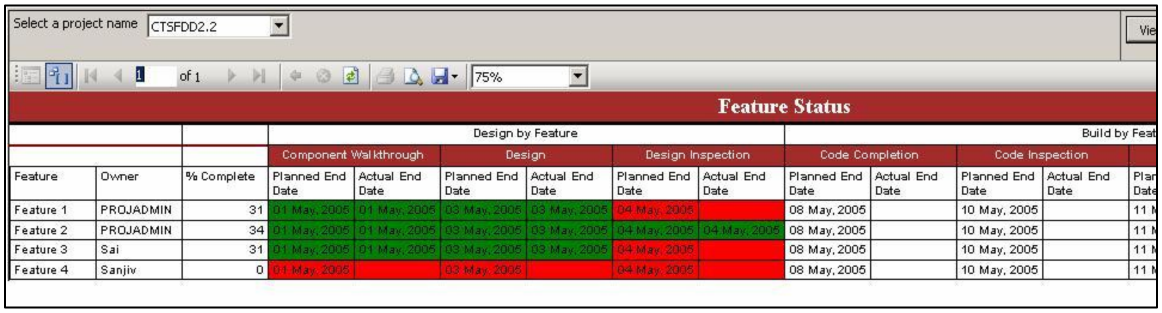
Checkin cannot proceed because the policy requirements have not been satisfied.

☐ Override policy failure and continue checkin

Reason:

Figure 26: Check-in Policy Enforcement in Build by Feature

The Project Manager will be able to generate the report and see how the features are progressing through the construction phase. A sample report is shown in the following figure. Note that in the FDD model, different features will be at different stages of completion at any given point in time.



Select a project name CTSFDD2.2

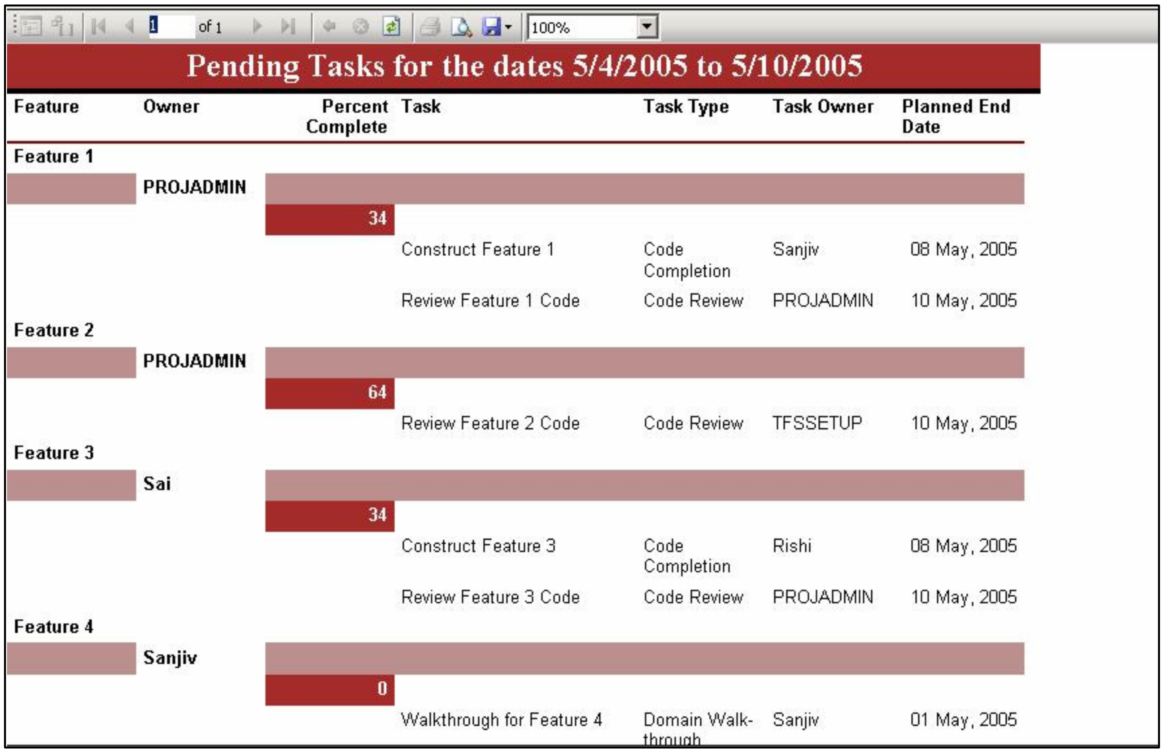
1 of 1 75%

Feature Status

			Design by Feature						Build by Feat				
			Component Walkthrough		Design		Design Inspection		Code Completion		Code Inspection		
Feature	Owner	% Complete	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Plan Date
Feature 1	PROJADMIN	31	🟩 01 May, 2005	🟩 01 May, 2005	🟩 03 May, 2005	🟩 03 May, 2005	🟥 04 May, 2005	🟥	08 May, 2005		10 May, 2005		11 M
Feature 2	PROJADMIN	34	🟩 01 May, 2005	🟩 01 May, 2005	🟩 03 May, 2005	🟩 03 May, 2005	🟩 04 May, 2005	🟩 04 May, 2005	08 May, 2005		10 May, 2005		11 M
Feature 3	Sai	31	🟩 01 May, 2005	🟩 01 May, 2005	🟩 03 May, 2005	🟩 03 May, 2005	🟥 04 May, 2005	🟥	08 May, 2005		10 May, 2005		11 M
Feature 4	Sanjiv	0	🟥 01 May, 2005	🟥	🟥 03 May, 2005	🟥	🟥 04 May, 2005	🟥	08 May, 2005		10 May, 2005		11 M

Figure 27: Feature Status Report

There can be various reports created for different purposes (such as the list of pending tasks across features as shown).



1 of 1 100%

Pending Tasks for the dates 5/4/2005 to 5/10/2005

Feature	Owner	Percent Complete	Task	Task Type	Task Owner	Planned End Date
Feature 1						
	PROJADMIN					
		34				
			Construct Feature 1	Code Completion	Sanjiv	08 May, 2005
			Review Feature 1 Code	Code Review	PROJADMIN	10 May, 2005
Feature 2						
	PROJADMIN					
		64				
			Review Feature 2 Code	Code Review	TFSSETUP	10 May, 2005
Feature 3						
	Sai					
		34				
			Construct Feature 3	Code Completion	Rishi	08 May, 2005
			Review Feature 3 Code	Code Review	PROJADMIN	10 May, 2005
Feature 4						
	Sanjiv					
		0				
			Walkthrough for Feature 4	Domain Walk-through	Sanjiv	01 May, 2005

Figure 28: Pending Tasks for a Time Period

One of the important reports in Cognizant FDD is the Planned vs. Actual feature completion. This clearly tells whether the project is on track or not. The custom report created in VSTS clearly demonstrates it as follows.

Cognizant Technology Solutions

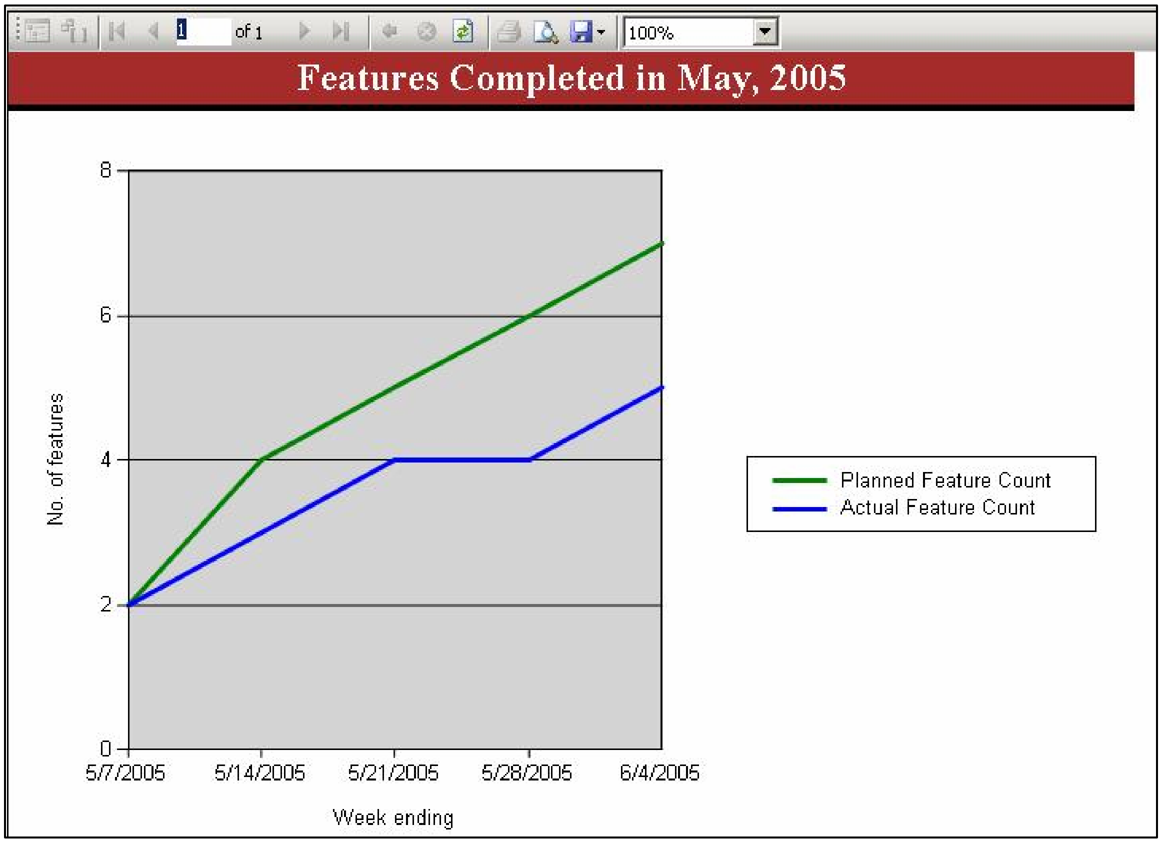


Figure 29: Features Planned vs. Actual Completed

Certification

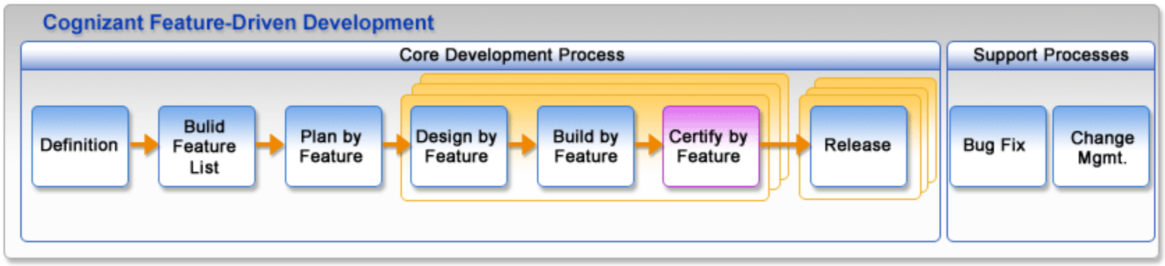


Figure 30: Certification Work Stream

This work stream is a per-feature activity to test feature functions as desired in both normal and unexpected circumstances. Every feature should have one or more test cases that demonstrate that feature. The Tester creates test cases for the feature. The Feature Owner reviews the test case created by the Tester. Once the review process is successful, the Tester creates automated test scripts. The Tester runs the automated test scripts on the build promoted in “build by feature” work stream.

The roles involved and their activities in this work stream are:

Tester:

- Create test cases for the feature.
- Create automated test scripts.
- Run automated test scripts on the implemented feature.
- Report defects if the test fails.

Feature Owner:

- Review test cases.

Project Manager:

- Monitor and track the progress of all these activities.
- Analyze the defects created by the Testers and assign them to the right person.

The following figure represents activities in the Certification work stream and the roles performing them.

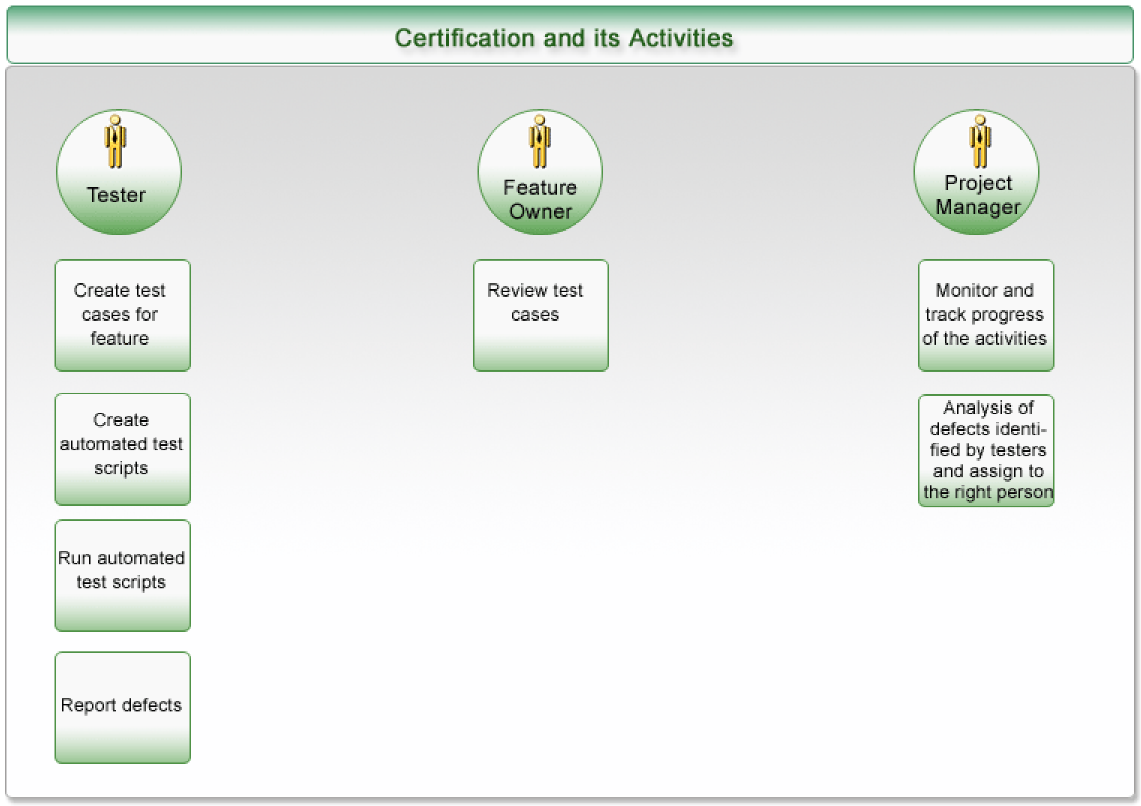
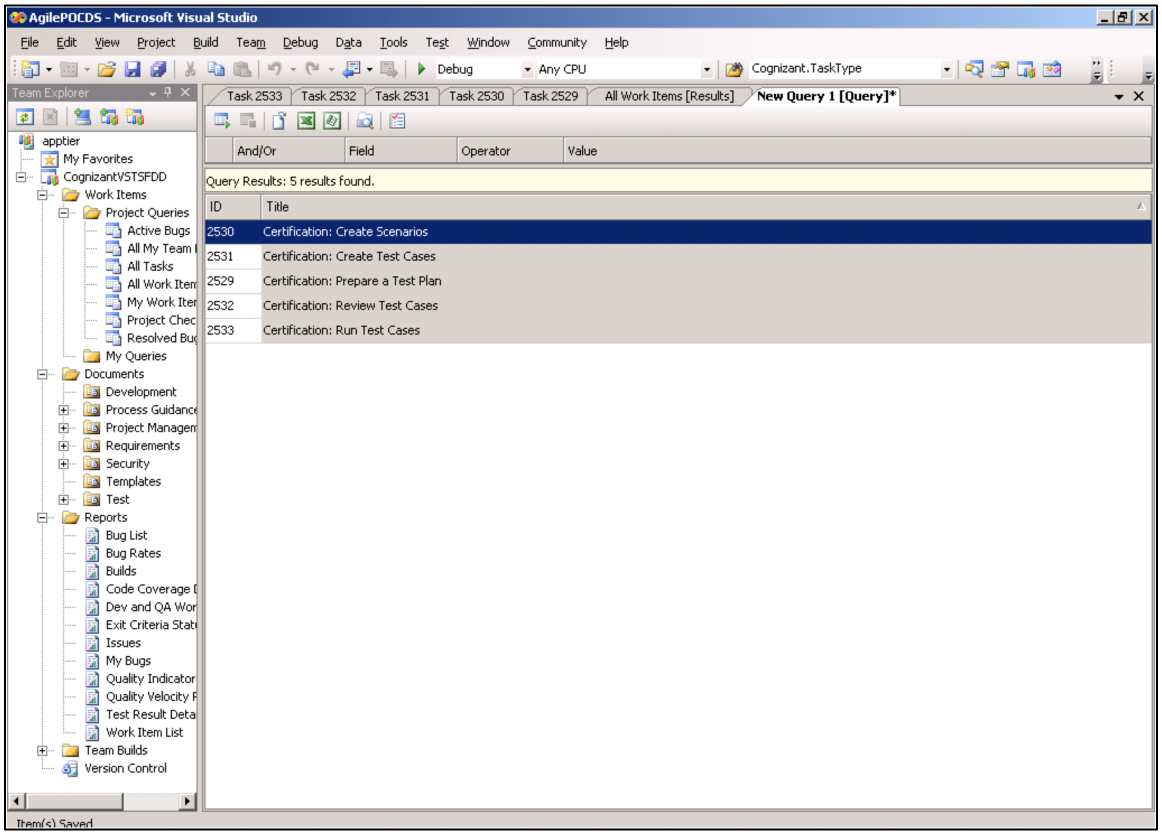

Certification and its Activities
Tester
Create test cases for feature
Create automated test scripts
Run automated test scripts
Report defects
Feature Owner
Review test cases
Project Manager
Monitor and track progress of the activities
Analysis of defects identi-fied by testers and assign to the right person

Figure 31: Role-based Activities in Certification Work Stream

The following figure shows the tasks that a Tester will view during the certification of a feature.

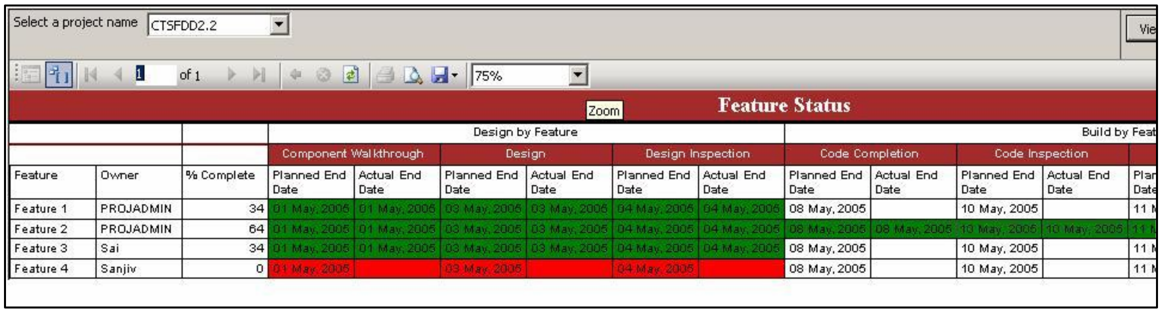


Query Results: 5 results found.

ID	Title
2530	Certification: Create Scenarios
2531	Certification: Create Test Cases
2529	Certification: Prepare a Test Plan
2532	Certification: Review Test Cases
2533	Certification: Run Test Cases

Figure 32: Certification Tasks for a Tester

The following status report is a snapshot of a project taken during certification. It shows that while one of the features is in the certification stage, the other features are still in other stages. Note that one of the features has missed the deadline and is shown in red.



Feature Status

			Design by Feature						Build by Feat				
			Component Walkthrough		Design		Design Inspection		Code Completion		Code Inspection		
Feature	Owner	% Complete	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Planned End Date	Actual End Date	Plan Date
Feature 1	PROJADMIN	34	01 May, 2005	01 May, 2005	03 May, 2005	03 May, 2005	04 May, 2005	04 May, 2005	08 May, 2005		10 May, 2005		11 M
Feature 2	PROJADMIN	64	01 May, 2005	01 May, 2005	03 May, 2005	03 May, 2005	04 May, 2005	04 May, 2005	08 May, 2005	08 May, 2005	10 May, 2005	10 May, 2005	11 M
Feature 3	Sai	34	01 May, 2005	01 May, 2005	03 May, 2005	03 May, 2005	04 May, 2005	04 May, 2005	08 May, 2005		10 May, 2005		11 M
Feature 4	Sanjiv	0	01 May, 2005		03 May, 2005		04 May, 2005		08 May, 2005		10 May, 2005		11 M

Figure 33: Feature Status Report

Release

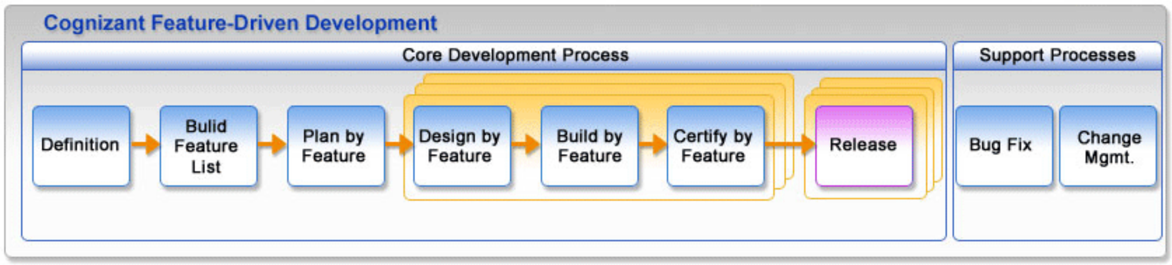


Figure 34: Release Work Stream

This work stream consists of activities involved in the release of one or more features. These features are grouped into releases based on various factors such as feature dependencies, client requirement, etc. Features identified can be across the feature sets and this grouping of features for release is independent of the feature sets defined earlier. Once the features have been identified, they have to be tested for integration with the rest of the application. Then an installation package is created with the features added in the release and an installation guide is created/ updated. The Testers run user acceptance tests on the new installation and finally release notes are written for the release.

The roles involved and their activities in this work stream are:

Project Manager:

- Identify the features that would go in a release and create tasks for the rest of the team.
- Create detailed release notes for this release.

Tester:

- Run integration tests for the features to be released.
- Run acceptance tests on the new installation package.

Developer:

- Create installation (deployment) package
- Write user installation guide

The following figure represents activities in the Release work stream and the roles performing them.

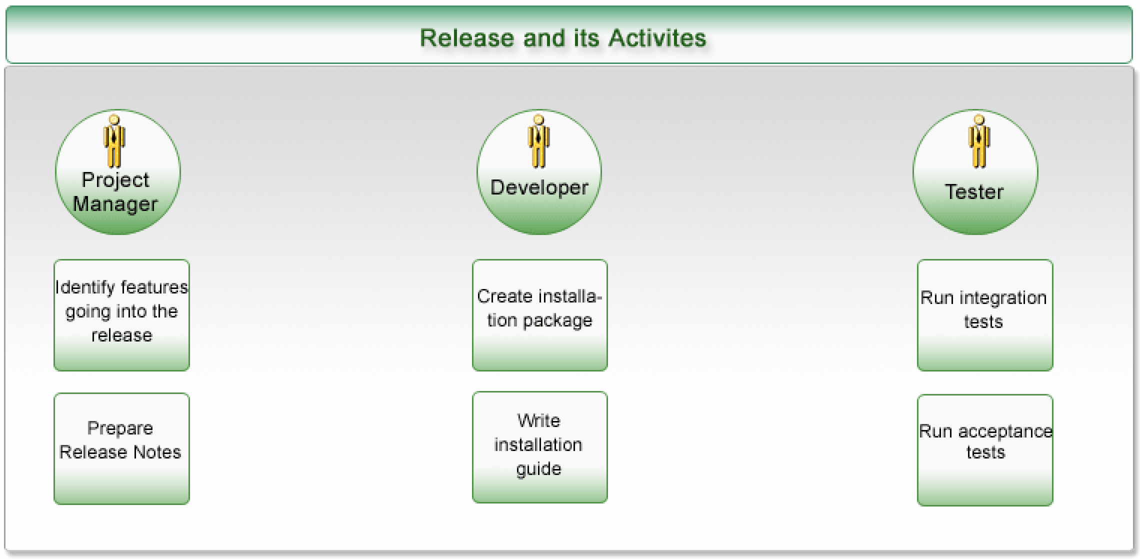
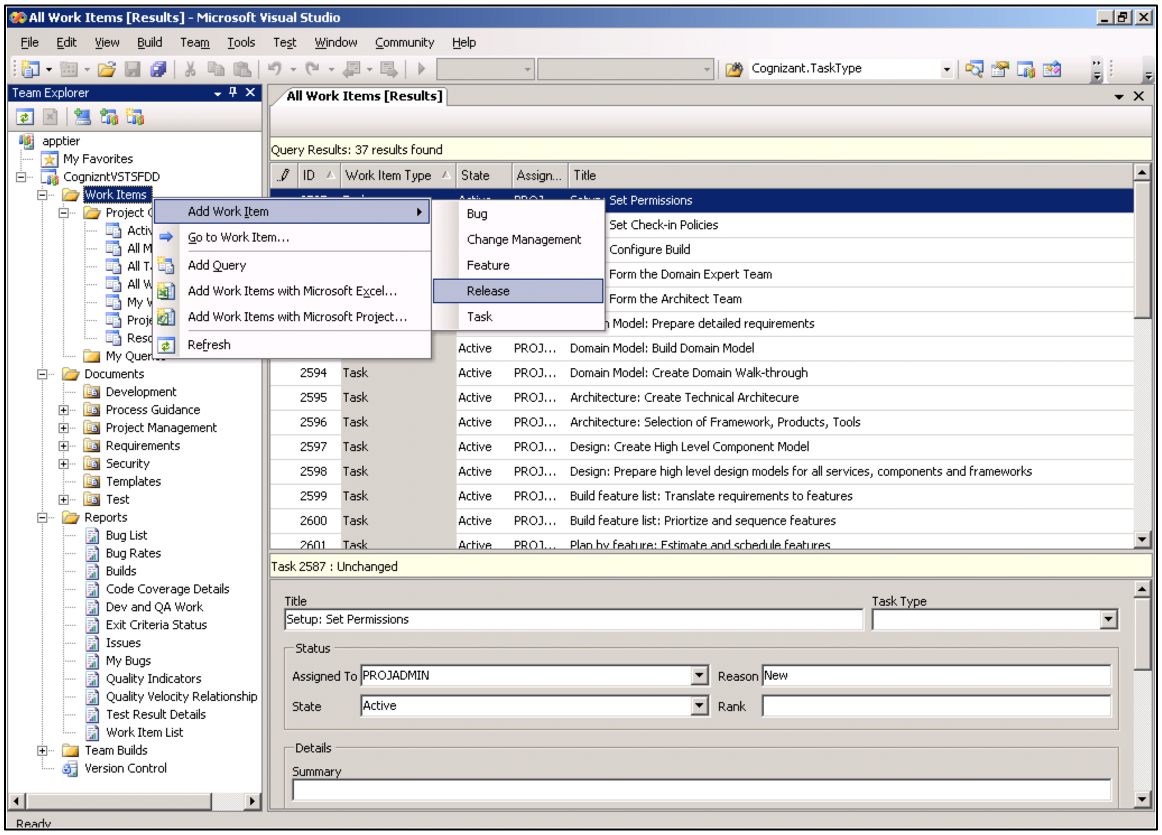


Figure 35: Role-based Activities in Release Work Stream

Release work items include tasks to identify features for a release, integration testing, creation of installation package, writing user installation guide and release notes. Creation of a release work item is shown in the following figure.



All Work Items [Results]

Query Results: 37 results found

ID	Work Item Type	State	Assign...	Title
				Set Permissions
				Set Check-in Policies
				Configure Build
				Form the Domain Expert Team
				Form the Architect Team
				Model: Prepare detailed requirements
		Active	PROJ...	Domain Model: Build Domain Model
2594	Task	Active	PROJ...	Domain Model: Create Domain Walk-through
2595	Task	Active	PROJ...	Architecture: Create Technical Architecure
2596	Task	Active	PROJ...	Architecture: Selection of Framework, Products, Tools
2597	Task	Active	PROJ...	Design: Create High Level Component Model
2598	Task	Active	PROJ...	Design: Prepare high level design models for all services, components and frameworks
2599	Task	Active	PROJ...	Build feature list: Translate requirements to features
2600	Task	Active	PROJ...	Build feature list: Priortize and sequence features
2601	Task	Active	PROJ...	Plan by feature: Estimate and schedule features

Task 2587 : Unchanged

Title: Setup: Set Permissions
Task Type:

Status
Assigned To: PROJADMIN
Reason: New
State: Active
Rank:

Details
Summary:

Figure 36: Creation of Release Work Items

Bug Fix

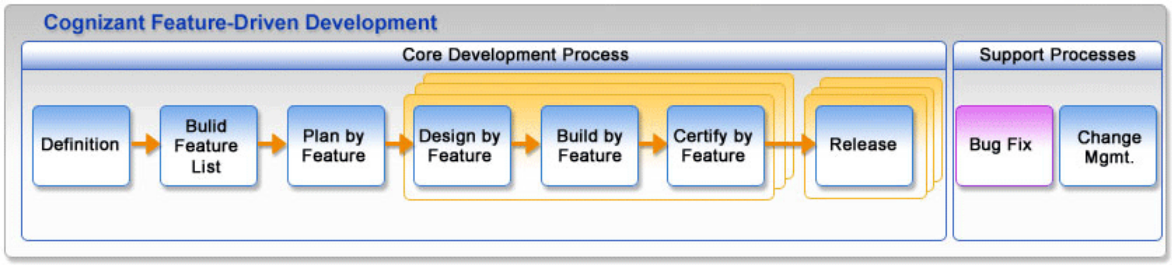


Figure 37: Bug Fix Work Stream

This work stream handles tracking of bugs recorded every time a test fails or as reported from the field. A fix for a bug could result in a change in "design by feature" and/or "build by feature" work stream. Based on which work stream the bug affects, the respective Designers or Developers would make fixes to their artifacts. The subsequent process would resemble similar activities in the affected work streams ("design by feature" or/and "build by feature").

The roles involved and their activities in this work stream are:

Project Manager:

- Assign, prioritize and track defects raised and bugs fixed.

Developer:

- Implement code fixes.
- Perform unit testing.

Designer:

- Revise design if the bug warrants.
- Review the Developers' fix.

Feature Owners:

- Analyze the features impacted by a bug.

Tester:

- Run automated test scripts for the fixes.

The following figure represents activities in the Bug Fix work stream and the roles performing them.

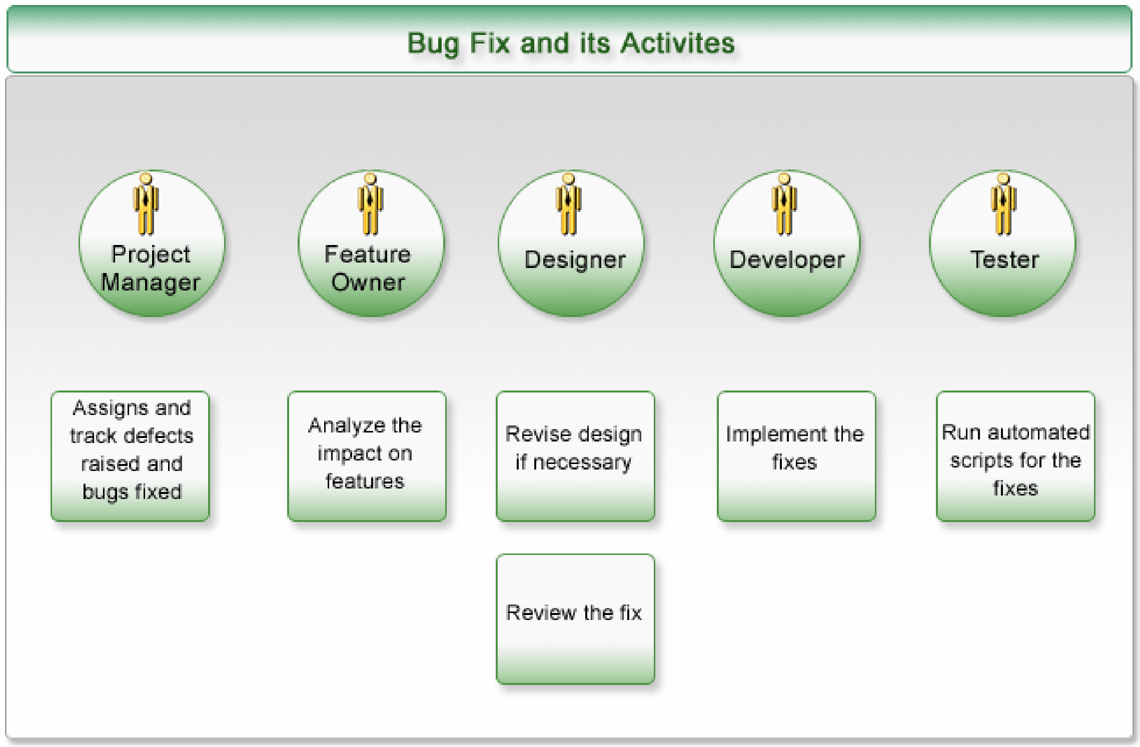
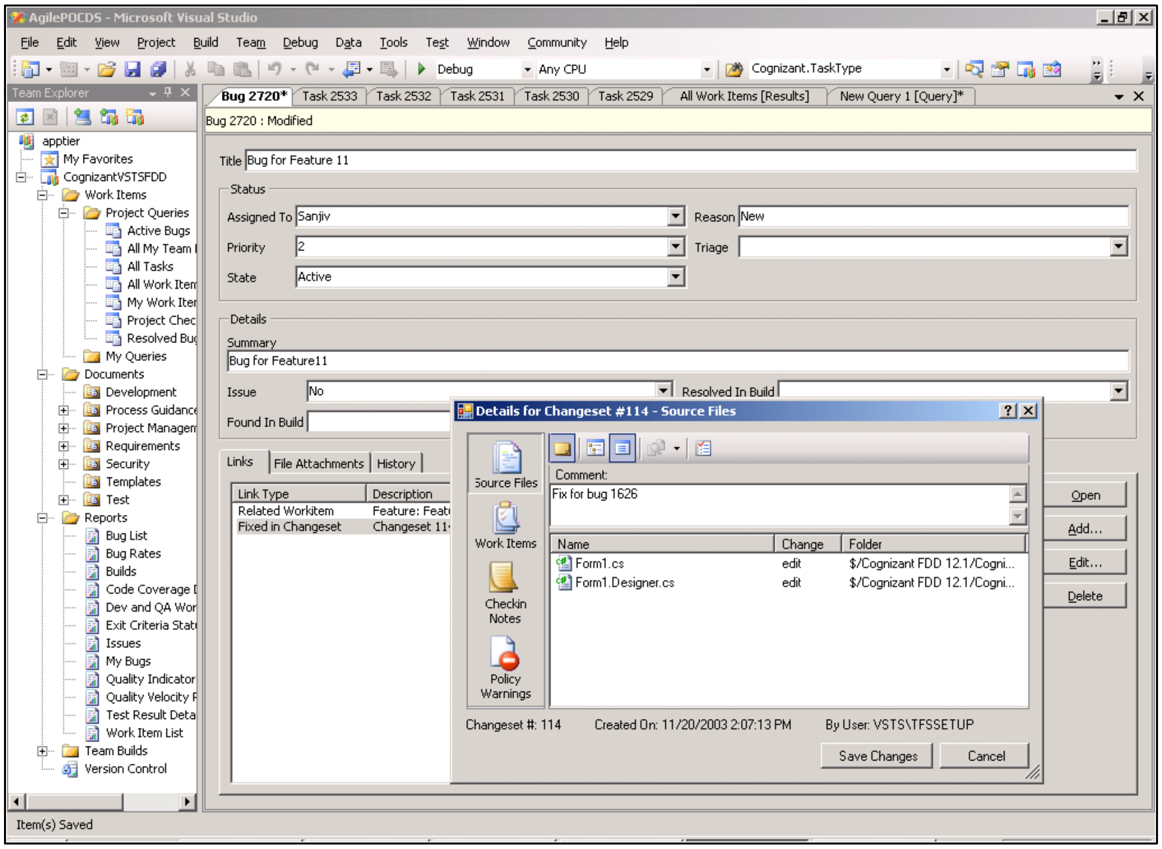

Bug Fix and its Activites
Project Manager
Feature Owner
Designer
Developer
Tester
Assigns and track defects raised and bugs fixed
Analyze the impact on features
Revise design if necessary
Implement the fixes
Run automated scripts for the fixes
Review the fix

Figure 38: Role-based Activities in Bug Fix Work stream

The code changes done for fixing a bug are grouped together and can be tracked as a change-set. The following figure shows the change-set being linked to the bug.



Bug 2720 : Modified

Title: Bug for Feature 11

Status

Assigned To: Sanjiv | Reason: New
Priority: 2 | Triage:
State: Active

Details

Summary: Bug for Feature11

Issue: No | Resolved In Build:
Found In Build:

Link Type	Description
Related Workitem	Feature: Feat
Fixed in Changeset	Changeset 11

Details for Changeset #114 - Source Files

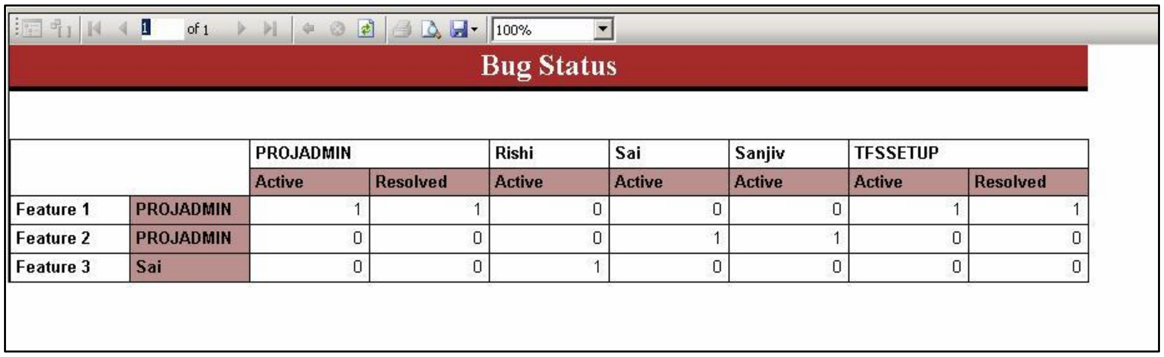
Comment: Fix for bug 1626

Name	Change	Folder
Form1.cs	edit	$/Cognizant FDD 12.1/Cogni...
Form1.Designer.cs	edit	$/Cognizant FDD 12.1/Cogni...

Changeset #: 114   Created On: 11/20/2003 2:07:13 PM   By User: VSTS\TFSSETUP

Figure 39: Fix for a Bug

One of the reports that the Project Manager can use at this stage is a bug status report, which is shown in the following figure. It shows bugs based on the feature and the Developer with the number of defects in active and resolved states.



Bug Status

		PROJADMIN		Rishi	Sai	Sanjiv	TFSSETUP	
		Active	Resolved	Active	Active	Active	Active	Resolved
Feature 1	PROJADMIN	1	1	0	0	0	1	1
Feature 2	PROJADMIN	0	0	0	1	1	0	0
Feature 3	Sai	0	0	1	0	0	0	0

Figure 40: Bug Status Report

Change Management

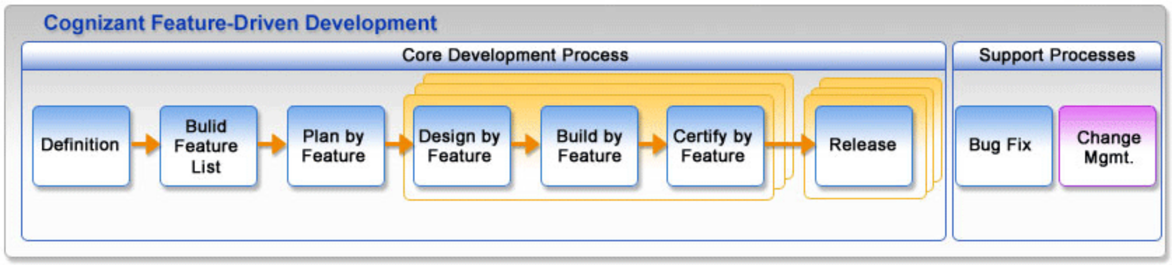


Figure 41: Change Management Work Stream

This work stream handles change requests, which can be initiated from the user or the client. Change requests may require modifications to features completed or currently in progress. Additions and changes to features initiated from a change request need to be tracked separately. The Project Manager has a vital role to play in this work stream. A change request can affect from any of the phases from "definition" to "build by feature" work streams. The Project Manager, with inputs from his team, decides on the severity and judiciously decides the extent of impact of the change request and assigns the identified role to do impact analysis. Once the impact analysis is done, changes to the affected feature/features would be implemented.

The roles involved and their activities in this work stream are:

Project Manager:

- Assign change request to an apt role based on experience to conduct impact analysis.
- Estimate change implementation.
- Plan change implementation.
- Monitor and track the progress of this activity.

Architect:

- Analyze impact on architecture if relevant for the change request.

Domain Expert:

- Analyze impact on domain model if relevant for the change request.

Developer:

- Change code implementation and perform unit testing relevant for the change request.

Designer:

- Revise design if relevant for the change request.
- Conduct code reviews for the changes implemented.

Feature Owners:

- Analyze features impacted.

Testers:

- Create and revise test cases if relevant for the change request.
- Update test scripts for changes and run automated scripts.

The following figure represents activities in the Change Management work stream and the roles performing them.

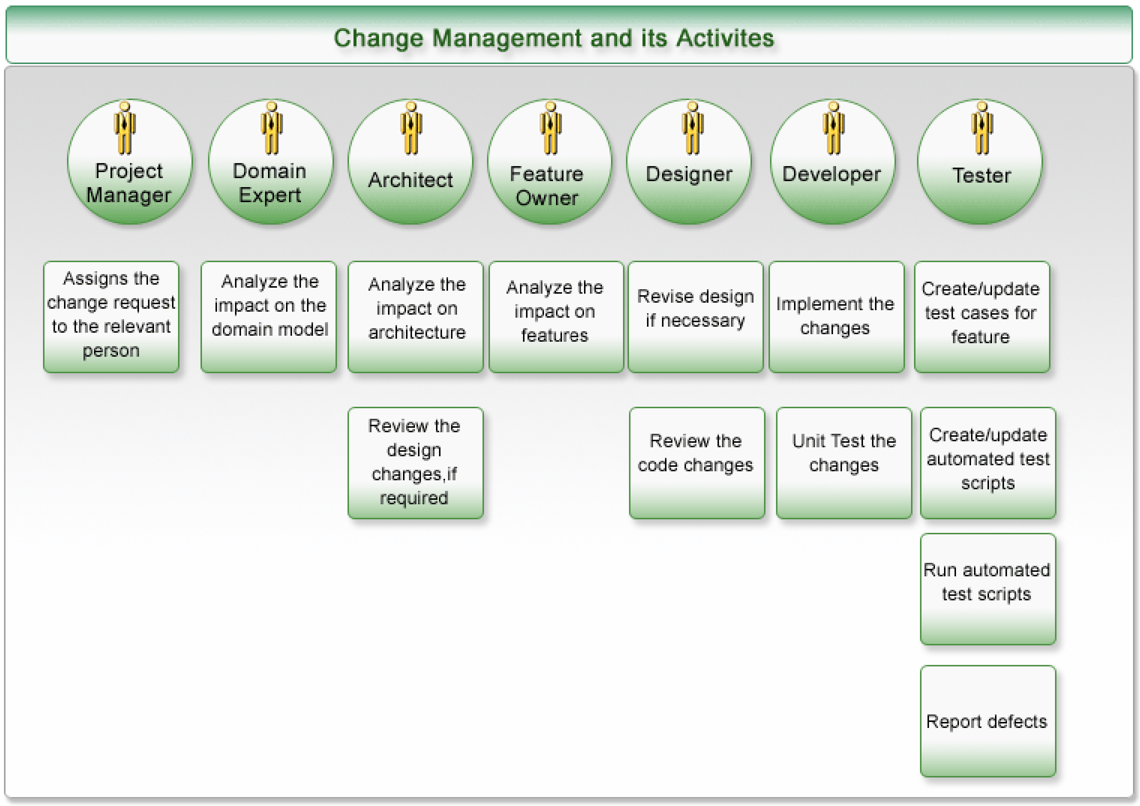
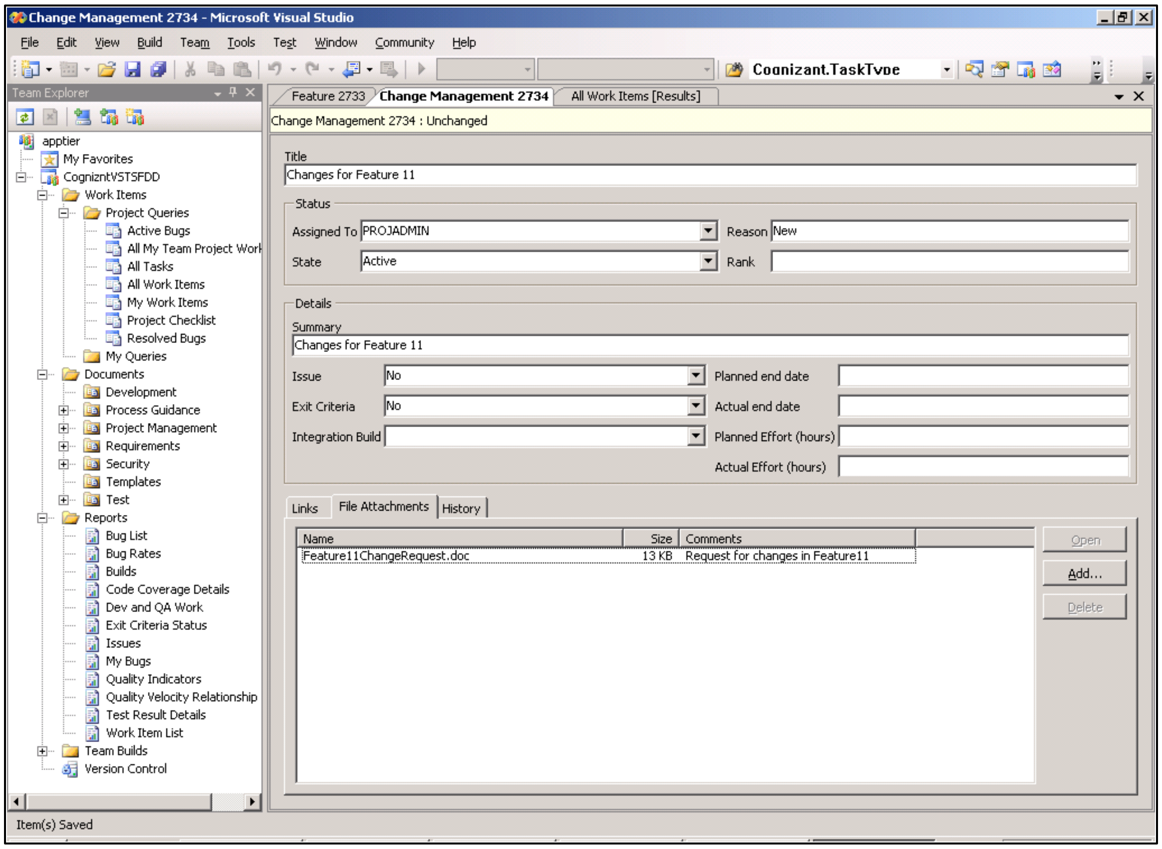

Change Management and its Activites
Project Manager
Domain Expert
Architect
Feature Owner
Designer
Developer
Tester
Assigns the change request to the relevant person
Analyze the impact on the domain model
Analyze the impact on architecture
Analyze the impact on features
Revise design if necessary
Implement the changes
Create/update test cases for feature
Review the design changes,if required
Review the code changes
Unit Test the changes
Create/update automated test scripts
Run automated test scripts
Report defects

Figure 42: Role-based Activities in Change Management Work Stream

The following figure shows a change request being linked to a feature change work item.



Change Management 2734 - Microsoft Visual Studio

Change Management 2734 : Unchanged

Title: Changes for Feature 11

Status

Assigned To: PROJADMIN
Reason: New
State: Active
Rank:

Details

Summary: Changes for Feature 11

Issue: No
Planned end date:
Exit Criteria: No
Actual end date:
Integration Build:
Planned Effort (hours):
Actual Effort (hours):

Name	Size	Comments
Feature11ChangeRequest.doc	13 KB	Request for changes in Feature11

Item(s) Saved

Figure 43: Changes for a Feature

Conclusion

Agile processes with automation and tool integration can scale up for large-scale business software development. Cognizant FDD integrated with Microsoft VSTS provides the automated process infrastructure for large-scale, feature-driven development that can enhance the productivity and quality with reduced overhead.

It is an extremely complex task to manage and track a number of features in a large-scale development across different feature teams, all affecting the same design and code base at the same time. Cognizant FDD integrated with VSTS provides a great degree of automation to manage the allocation, tracking and implementation of different features across different roles.

VSTS is also integrated with the rest of the Visual Studio tools so that processes and artifacts are completely tied up. This increases the productivity drastically and ensures that the whole team completely adheres to the process and the process is highly automated ensuring less time per feature creation.

Above all, Cognizant FDD implementation further proves that it is easy to implement a custom development process using VSTS.

Cognizant Technology Solutions

About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of IT services focused on delivering strategic information technology solutions that addresses the complex business needs of its clients.

Cognizant provides applications management, development, integration, re-engineering, infrastructure management, business process outsourcing, and a number of related services such as enterprise consulting, technology architecture, program management and change management through its onsite/offshore outsourcing model.

Cognizant has strong expertise in implementing enterprise applications on the Microsoft .NET platform, with almost a third of it's over 18,000 employees working on Microsoft technology projects. Cognizant has been recognized as a pioneer in using agile development methodologies in offshore outsourcing. Cognizant is a proven leader in delivering large, complex applications for Fortune 500 clients using iterative and agile processes.

Cognizant Technology Solutions
500 Glenpointe Centre West
Teaneck, NJ 07666
Ph: 201-801-0233
Fax: 201-801-0243
Toll free: 1-888-937-3277
www.cognizant.com