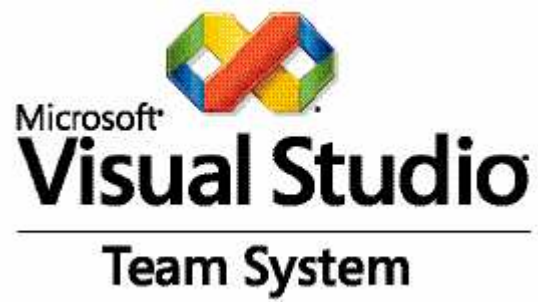


CAPÍTULO II

VISUAL STUDIO TEAM SYSTEM



2.1. INTRODUCCIÓN

Visual Studio 2005 Team System, nombre en clave Burton, aporta un conjunto de herramientas dirigidas a desarrolladores, arquitectos, administradores de proyectos, testers, administradores de base de datos en definitiva, al equipo de desarrollo de una aplicación. Para los que conocen bien el desarrollo con Borland¹³, Visual Studio Team System es una herramienta ALM, similar a la iniciativa de Borland para facilitar dicho proceso de desarrollo.

Dando un punto de vista objetivo se mostrará las perspectivas de la siguiente manera:

2.1.1. ADMINISTRADOR DE PROYECTOS

La principal preocupación es obtener información real y objetiva acerca del estatus del proyecto, es de suma importancia medir la velocidad del desarrollo, el número y el impacto de las deficiencias encontradas, los riesgos, entre otros datos importantes. Sin embargo es difícil recolectar y analizar esta información además sin decir que esta información tenga una veracidad importante.

Manejar un Proyecto de Software sin información de los defectos es cómo manejar un auto con los ojos vendados y solo cambiar de dirección cuando golpeas contra algo^[11].

2.1.2. PROBANDO Y DESARROLLANDO

Es de suma importancia entender y comprender los conceptos como pruebas unitarias, cobertura de código, análisis estático y rendimiento, sin embargo este tipo de análisis y pruebas que no son solo responsabilidad del equipo de pruebas conlleva el uso de herramientas que si bien están diseñadas con ese

¹³ Borland. Es una compañía de software, ubicada en Scotts Valley, California, Estados Unidos, conocida por sus herramientas de programación, especialmente Turbo Pascal que evolucionó hasta el actual Delphi.

^[11] Página xxi, Software Engineering with Microsoft Visual Studio Team System – Sam Guckenheime.

objetivo no están integradas al ambiente de desarrollo y sin importar lo difícil que pueda ser usarlas se pierde agilidad en el proceso de desarrollo, como una solución para ello, Visual Studio Team System a integrado herramientas para probadores de software, las cuales hacen más ágil y efectivo este proceso.

2.1.3. ARQUITECTO DE SOFTWARE

Pueden ser muy importantes, atractivos e impresionantes los modelos de aplicación y despliegue , sin embargo a pesar de esto, llevar a cabo este planteamiento que se describe en cada modelo es una tarea muy difícil ya que principalmente el tiempo que toma se traduce en un costo alto, incluso para los equipos de desarrollo más experimentados. Además de que no siempre se plasman todos los detalles que son importantes para el proceso de desarrollo y pruebas.

Después de dar una vista muy rápida a algunas partes que conforman el proceso de desarrollo, la pregunta es como poder llevar a cabo el desarrollo de una solución sin tener que sacrificar elementos que después serán costos.

2.2. ANTECEDENTES

Con Visual Studio .NET y la plataforma .NET Framework, Microsoft empezaba una nueva era, apoyando y facilitando la migración y la interoperabilidad con COM¹⁴, sabiendo que .NET¹⁵ iba a ingresar, poco a poco, en muchas empresas y departamentos TI. Su rápida expansión y el desarrollo de aplicaciones distribuidas así como SOA¹⁶ hacían que grandes proyectos resultaran excesivamente caros a la hora de sacar la máxima productividad durante su ciclo de vida.

¹⁴ COM. (Component Object Model) para soportar comunicación entre objetos en ordenadores distintos, en redes de distinto alcance, o incluso en Internet.

¹⁵ .NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

¹⁶ SOA. Arquitecturas basadas en servicios.

Con la aparición de Visual Studio .NET 2005, Microsoft ofreció Visual Studio Team System que corrige todas las deficiencias permitiendo que el alto crecimiento de la complejidad de las aplicaciones y el ciclo de vida de dicha aplicación, sea cual sea su fase, estén respaldados por una serie de herramientas y guías necesarias que permitan resultados fiables, sin poner en riesgo ni la productividad ni la escalabilidad de la aplicación^[12].

Basándose en la demostrada experiencia de Microsoft en la creación de entornos de desarrollo, esta versión expande significativamente su alcance y productividad ofreciendo herramientas de ciclo de vida completamente integradas y extensibles para incrementar la previsibilidad del proceso de desarrollo del software.

En la actualidad el desarrollo de software donde se implementan varias metodologías, desde las ágiles como podría ser Xtreme Developing¹⁷ o MSF¹⁸ for Agile, hasta las tradicionales donde el desarrollador de software era el responsable total del desarrollo. Se han integrado nuevos roles en el proceso de llevar a cabo una solución integral a las problemáticas de TI, basándonos en MSF for Agile estos roles van desde el Desarrollador de Software, pasando por el Ingeniero de Pruebas hasta el Arquitecto y el administrador de proyecto, estos roles tienen objetivos específicos y por lo tanto intereses particulares. Para un desarrollo exitoso utilizando dicha metodología se necesita que las actividades de cada uno de estos roles se lleven a cabo exitosamente^[13].

El principal gran cambio desde que Microsoft introdujo al mercado Visual Studio .Net 2002, fue el lanzamiento de Visual Studio Team System, con lo que se busca transformar esas problemáticas específicas de cada Rol en sus virtudes dando así un proceso integrador, ágil y con predictibilidad. Esta propuesta se puede describir como un conjunto de componentes donde se buscará transformar lo que ahora se puede ver como problemática a virtud. Este conjunto de componentes son:

^[12] <http://www.dotnetmania.com/Articulos/008/editorial.html>.

¹⁷ Xtreme Developing. Desarrollo extremo

¹⁸ MSF. Es una metodología para el desarrollo de software para la planificación, desarrollo y gestión de proyectos tecnológico.

^[13] <http://www.abox.com/productos.asp?pid=785>.

- Visual Studio 2005 Team Edition para Desarrolladores de software
- Visual Studio 2005 Team Edition para Arquitectos de Software.
- Visual Studio 2005 Team Edition para Probadores de Software.
- Visual Studio 2005 Team Edition para profesionales de Base datos.
- Visual Studio 2005 Team Foundation Server.

2.3. FACTORES RELEVANTES DE MVTS.

Es el conjunto de herramientas con las cuales se puede optimizar el proceso de desarrollo de una solución, sacando ventaja de su integración al ambiente de desarrollo Visual Studio 2005, dando así un modelo diferente para cubrir las necesidades de cada Rol en particular.

Dando una vista rápida podremos decir que entre sus principales características serian:

a) Integración con Visual Studio 2005

Siempre es importante la resistencia del equipo de desarrollo al cambio, las nuevas políticas nunca son agradables sobre todo si se lleva mucho tiempo creando soluciones de alguna manera en particular.

Visual Studio Team System incrementa la productividad y consistencia de las aplicaciones o proyectos integrándose junto la interfaz de usuario, con los datos y con los procesos.

En la integración de procesos, el comportamiento de la herramienta de Team System es dictado por el propio proceso del proyecto. De esta manera se asegura que ningún tipo de información se pierda entre las distintas fases del proyecto o entre varios grupos de equipos.

b) Facilidad de Uso

Dada la integración mencionada anteriormente se reduce considerablemente la curva de aprendizaje en la utilización de las herramientas proporcionadas, algo que hasta el momento era difícil encontrar en el mercado. Dado este hecho se pueden reducir los tiempos en que se llevan a cabo estas actividades.

c) Extensibilidad

Cada persona en el equipo de trabajo es diferente, por lo tanto cada equipo de trabajo es diferente y cada empresa es diferente. Por lo que se necesita la capacidad de adaptar esta solución a las necesidades particulares de cada empresa, equipo y persona, para que así pueda llevarse una completa integración y rápida implementación.

d) Seguridad

Todas las soluciones creadas hoy en día deben de mantener estrictos márgenes de seguridad, dado este punto también deberíamos pensar que nuestro manejo de proyectos, código, pruebas, modelos, entre otros debería tener así una confiabilidad.

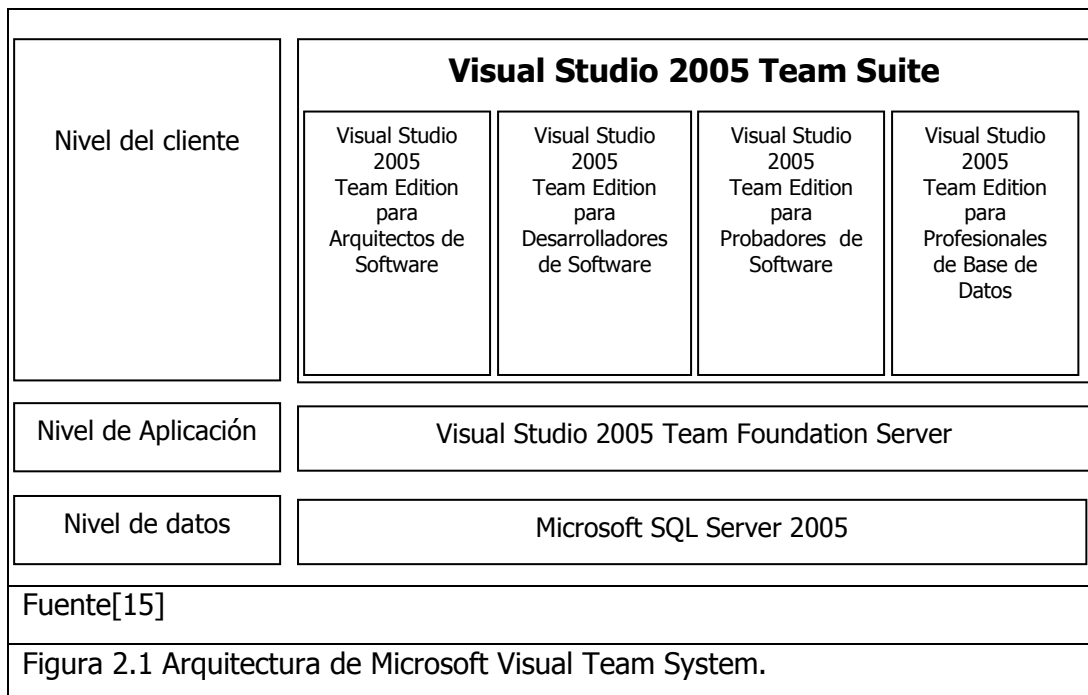
Implementando seguridad a varios niveles, con un conjunto de permisos por proyecto por ejemplo. Visual Studio Team System logra brindar la confiabilidad necesaria de que la información solamente llegara a la persona indicada^[14].

2.4. ARQUITECTURA DE VISUAL STUDIO TEAM SYSTEM

Visual Studio Team System es la nueva incorporación de Microsoft a la familia de Visual Studio. Se trata de un producto de gran importancia, ya que, por primera

^[14] [www.aquora.net/studentpartners/docs/Juan Jose Karam Visual Studio Team System.doc](http://www.aquora.net/studentpartners/docs/Juan%20Jose%20Karam%20Visual%20Studio%20Team%20System.doc)

vez, Microsoft lanza al mercado un producto dirigido a todo el ciclo de vida del desarrollo de software. Basado en el conjunto de características de Microsoft Visual Studio 2005, Team System incluye características adicionales que facilitan el trabajo de los arquitectos, desarrolladores, comprobadores y jefes de proyecto. Visual Studio 2005 Team System le proporcionará una perspectiva de extremo a extremo sobre las características más importantes del producto. Uno de los aspectos más importantes de Team System es que está diseñado para adaptarse a sus necesidades y no al revés. A diferencia de otros productos de Microsoft, como Office, Team System no está diseñado para un uso inmediato. Tiene que tomarse algo de tiempo para configurarlo y aplicarlo a sus directivas y prácticas, básicamente, modificarlo a la forma en la que usted desarrolla el software. Si su enfoque no está demasiado definido, Team System le ofrecerá un buen framework¹⁹ para diseñar un proceso adecuado y en el que se pueda trabajar Team System está compuesto de varios niveles lógicos. La figura 2.1 ofrece una vista general de la arquitectura de Team System.^[15]



¹⁹ Framework. Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

^[15] Profesional Visual Studio 2005 Team System de Jean-Luc, Tony Loton, Erick Gunvaldson, Christopher Bowen, Noah Coad, Darren Jefford Pág. 30 y pág. 31.

2.4.1. NIVEL DEL CLIENTE

El nivel del cliente está compuesto de varias versiones basadas en funciones de Visual Studio 2005 y otras herramientas que están diseñadas para conectar los niveles de datos y de la aplicación de Team System. A continuación se describe los cuatro productos del cliente de Team System:

2.4.1.1. VISUAL STUDIO 2005 TEAM SUITE.

Incorpora en un programa las funciones de Visual Studio 2005 Team Edition para Arquitectos de Software, Visual Studio 2005 Team Edition para Desarrolladores de Software y Visual Studio 2005 Team Edition para Probadores de Software. Este producto está dirigido a miembros de equipo que deseen llevar a cabo varias funciones. Team Suite resulta además útil para instalar en el servidor de Team Foundation Build, para incorporar las funciones de pruebas tanto de las versiones de desarrolladores como las de probadores del producto.

a. Visual Studio 2005 Team Edition para Arquitectos de Software.

Incluye herramientas para soportar los arquitectos de la aplicación y de la infraestructura, durante la fase de diseño inicial y el desarrollo en curso de un proyecto de software, incluye varios diseñadores, como las herramientas para modelar la aplicación, modelar la infraestructura y modelar la implementación. No hay que confundir esta herramienta con Visual Studio .NET 2003 para Arquitectos. Esta se consideraba la versión más completa de Visual Studio. NET 2003. La herramienta equivalente es Visual Studio 2005 Team Suite.

b. Visual Studio 2005 Team Edition para Desarrolladores de Software.

Cuenta con analizadores de código estático tanto para código administrado como para código no administrado, analizadores de código

dinámico, conocidos también como App Verifier, cobertura de pruebas unitarias y herramientas de generación de perfiles de código.

c. Visual Studio 2005 Team Edition para Probadores de Software.

Tiene funciones como la administración de casos de prueba, las pruebas de carga y de rendimiento, las pruebas unitarias, la cobertura de código y las pruebas manuales.

d. Visual Studio 2005 Team Edition para Profesionales de base de datos.

Es un nuevo rol en el Ciclo de Vida de Desarrollo de Visual Studio Team System, tiene herramientas que ayudan a los desarrolladores y administradores de bases de datos a gestionar cambios, pruebas e implementación de bases de datos SQL Server, así como a crear aplicaciones relacionadas con bases de datos.

- Dispone de funciones específicas centradas en los módulos de gestión de datos en aplicaciones.
- Dispone de administradores de bases de datos que deben ser integrados en el proceso de desarrollo.

2.4.2. NIVEL DE APLICACIÓN.

Team Foundation Server incorpora los niveles de aplicación y de datos. Este servidor dispone de varias funciones integradas y personalizadas, como las funciones de administración de proyectos, en un sitio Web del portal del proyecto de equipo basado en Share Point²⁰, seguimiento de elementos de trabajo y generación de informes, automatización de la generación y un sistema de control de versiones. Team Foundation Server proporciona también soporte para seguridad, directivas y otras funciones administrativas.

²⁰ Sharepoint. Microsoft SharePoint es un grupo de tecnologías complementarias que hacen que compartir la información sea más fácil en Internet.

2.4.3. NIVEL DE DATOS.

El nivel de datos de Team System se basa en Microsoft SQL Server 2005. Este producto está especialmente seleccionado para reducir la complejidad de la aplicación y acelerar el tiempo de la comercialización. En este momento, Team System no es compatible con orígenes de datos de terceros y no está integrado con el servicio de datos empresariales. El nivel de datos de Team System se utiliza para guardar todos los datos de Team System, como código fuente, elementos de trabajo, datos de informes y ajustes de proyecto.

2.5. VISIÓN DE LOS ROLES

Dentro de la visión de roles se detallará los diferentes actores que intervienen en la administración de las herramientas que están integradas dentro de MVSTS.

2.5.1. ADMINISTRADOR DE PROYECTOS

“Es imposible que una metodología sea la correcta, ya existe una manera diferente de trabajo para cada proyecto y para cada equipo”^[16]

Visual Studio Team System se enfoca particularmente en el requerimiento de información veraz y oportuna que necesita particularmente este rol. Debemos comprender que una buena decisión no siempre es la salvación de un proyecto sobre todo si esta fue tomada demasiado tarde.

Utilizando un sistema de reporte esta información es presentada de manera entendible y a tiempo para una mejor toma de decisiones. El problema ha sido siempre la manera en que esta información es recolectada, mediante un conjunto de características tales como Workitems²¹, cobertura de código o pruebas unitarias se puede garantizar información veraz a un grado de detalle sumamente impresionante.

Similarmente a la realidad que tan difícil podría ser escribir a mano una carta con la mano izquierda cuando por nacimiento eres diestro, es el mismo

^[16] http://alistair.cockburn.us/index.php/The_declaration_of_interdependence_for_modern_management.

²¹ Workitems. Elemento de trabajo que se definen en Team System.

concepto que maneja Visual Studio Team System, que tan difícil puede ser crear Software de calidad cuando la metodología que tus aplicaciones manejan, no es la misma que la que utilizas. Este problema es sumamente fácil de resolver ya que aunque de fábrica solamente se incluyan dos metodologías como MSF for Agile y MSF for CMMI²², se tiene la capacidad de crear y personalizar la metodología a utilizar en cada uno de los proyectos, adaptando las herramientas al equipo, no el equipo a las herramientas.

2.5.2. ARQUITECTO DE SOFTWARE

Cada sistema tiene una arquitectura, abarcando las abstracciones claves y los mecanismos que definen el comportamiento y la estructura de dicho sistema según lo considerado de la perspectiva de diversos tomadores de decisiones, cada uno con un diverso conjunto de preocupaciones. En todos los casos estos patrones pueden ser intencionales o accidentales, a medida de que son vistos, esos patrones reflejan el estilo interno de cada sistema.^[17]

El objetivo de una arquitectura es proveer un modelo integral, fácil de entender, usar, mantener y extender a las necesidades del negocio. Sin embargo esta se vuelve una tarea bastante compleja dado que no siempre es sencillo implementar dichos modelos tanto en tiempo de desarrollo como en implementación.

Desde el momento en que se diseña la arquitectura de una aplicación puede verse la dificultad que traerá que esta se consuma con éxito. Desde los inicios de su implementación en desarrollo puede observarse que no son completamente cubiertos los requerimientos o incluso cambian, dadas las circunstancias que se presentan en este proceso. También se puede cometer un error grave al no prever que la infraestructura donde será montada dicha solución no es la correcta o incluso no cubre algún requerimiento crítico para el éxito de la misma.

²² CMMI. Capacidad de madurez del modelo de integración
^[17] <http://www.agileshift.cl/Tutorial/DesarrolloAgilParte1.pdf>

Visual Studio Team System provee herramientas que permitirán que el Arquitecto de Software pueda visualizar, prever e implementar su modelo con éxito, estas herramientas de diagramación permiten especificar desde el inicio del proyecto los requerimientos de Software, a un grado tal de definir las interfaces de servicio que contendrá un Servicio Web, los permisos necesarios de ejecución, la versión del .Net Framework a utilizar, entre otras más. También ayudara a prever que la infraestructura cumpla los requerimientos de la aplicación permitiendo así predecir posibles problemas en el momento de la implementación en producción.

Estos diagramas permiten de manera rápida poder entender los componentes que serán parte de la solución y visualizar su implementación en la infraestructura mencionada. Mostrando desde un inicio los requerimientos críticos para el éxito de la solución.^[18]

2.5.3. DESARROLLADOR

El desarrollo de una pieza de software puede ser por sí misma un reto complicado, este puede alcanzar un grado de complejidad mayor si tomamos en cuenta algunos problemas comunes que se presentan en los equipos de desarrollo.

En muchas ocasiones la falta de información puede llevar a errores críticos, dado que un requerimiento mal entendido o mal comunicado puede significar tiempo perdido, lo cual puede llevar al proyecto a no terminar a tiempo. Siempre es difícil definir el conjunto de tareas que realizara el desarrollador de software y más si estas están destinadas a cambiar o transformarse durante la marcha. Otro problema puede ser ocasionado por una mala comunicación con el equipo de pruebas, llevando así a una dificultad en la corrección de las imperfecciones y deficiencias en el código, siendo así una

^[18] <http://www.acis.org.co/memorias/JornadasGerencia/IIJNGP/AgileManifesto.pdf>

dura actividad, no el resolver el problema, si no entender y saber donde se localiza el mismo.

Sin embargo este no es el único problema al que se enfrenta el equipo de desarrollo, otra dificultad a resolver es el versionamiento de la solución, los cambios son constantes y en muchos casos piezas de la solución son alteradas al mismo tiempo por dos o más desarrolladores, lo que puede llevar a desconocer cuál es la versión más reciente o la ultima que fue probada con éxito.

Estos problemas son resueltos mediante algunas herramientas que provee Team System, mediante los Workitems se puede asignar a tiempo a el equipo de desarrollo un nuevo requerimiento y también así una imperfección, volviendo así la comunicación entre miembros del equipo algo ágil y confiable. Provee también herramientas de análisis de código y pruebas unitarias que proporcionaran información útil al equipo de cuanto porcentaje del código ha sido probado, cuando fue su última modificación e incluso regresar a una versión anterior.

El source control permite mantener un versionamiento y trazabilidad sobre todo el código, permitiendo así obtener una transparencia sobre el mismo, definiendo de manera rápida cuestiones como el último cambio, cual fue el mismo y quien lo produjo. Además pueden definirse reglas para que el código ingrese al Source Control²³, como pueden ser que un alto porcentaje del código haya sido probado o incluso que este cumpla con algún requerimiento o tarea asignada.

2.5.4. TESTER

Los proyectos son como un viaje en automóvil. Algunos proyectos son simples, como manejar en un día soleado por la carretera. Pero la mayoría de los proyectos son más parecidos a conducir por las montañas en una

²³ Source Control. Fuente de Control donde se guarda el código fuente, es solo una parte más de un team project (obviamente, con el tiempo, es la más grande y modificada).

camioneta de noche. Los Ingeniero de Pruebas, alumbran el camino. Iluminando por delante para que los desarrolladores y líderes, puedan saber donde están y que tan cerca están del barranco. Los objetivos del equipo de pruebas varían de proyecto a proyecto, sin embargo siempre hay un factor en común, las pruebas se realizan para encontrar información. Decisiones importantes sobre los proyectos se llevan a cabo basándose en esa información.^[19]

Obtener información nunca será tarea fácil y la necesidad de la misma siempre estará presente, en muchas ocasiones la capacidad de responder preguntas básicas sobre el estado del proyecto se convierte en una tarea sumamente complicada.

El objetivo del Testing no es soportar las actividades de desarrollo como podría llegar a pensarse en algún momento, las pruebas unitarias entre otras pruebas corresponden directamente al equipo de desarrollo. El objetivo principal del equipo de pruebas es proporcionar esa información vital para la toma de decisiones, esta deriva directamente del cumplimiento de los requerimientos de calidad, dichos requerimientos pueden ir desde Seguridad hasta su funcionamiento.

Muchos problemas son enfrentados cuando se realizan las pruebas en una solución, desde saber si todo el código ha sido probado, si se han realizado pruebas a los cambios, si los requerimientos de calidad han sido cumplidos satisfactoriamente y como más importante es saber si la solución funciona correctamente en ambiente de producción.

Visual Studio Team System ayuda a responder estas preguntas con información realista, permitiendo al Ingeniero de Pruebas realizar desde pruebas manuales hasta pruebas de carga, pero por si mismos estos resultados no significan mucho, así que también proporciona herramientas de análisis y diagnostico para así poder determinar cuáles son los riesgos,

^[19] http://thesharpcode.com/?page_id=49

problemas y deficiencias de dicha solución. Entregando así información consistente y veraz para la toma de decisiones.

2.6. VISIÓN DE LOS COMPONENTES

La visión de componentes de MVSTS expone las diferentes herramientas que se encuentran integradas, las cuales se detallan a continuación.

2.6.1. VISUAL STUDIO 2005 TEAM FOUNDATION SERVER

Es el gran repositorio de información del equipo donde se almacena toda la información como podría ser el código, la cantidad del mismo que ha sido probado, sin embargo no serviría de mucho contener este tipo de información si no se tiene la capacidad de explotarla correctamente.

Team Foundation Server utiliza otros componentes de Microsoft, los cuales son SQL Server 2005, Sharepoint y Reporting Services, con el objetivo de brindar las siguientes funcionalidades:

a) Control de Repositorio

Las fuentes de la solución ya no se encuentran en un repositorio físico como podrían ser carpetas del disco, ahora son almacenadas en una base de datos de SQL Server, lo cual beneficia en confiabilidad de los archivos y además proporciona información importante como puede ser el versionamiento y la trazabilidad.

b) Elementos de trabajo o WorkItem Tracking²⁴

Los workitems son elementos que se utilizan durante todo el proceso de desarrollo, entre los más comunes podemos definir una tarea, requerimiento o imperfección. Dichos workitems permiten la asignación fácil y rápida de tareas, el flujo dinámico de información entre los equipos

²⁴ WorkItem Tracking. Tipos de elementos de trabajo y definiciones de consulta.

que conforman el proyecto y muestran información útil que permite definir elementos como proyecto, velocidad de manera confiable.

c) Reportes

Una característica importante es la capacidad de explotar la información que es recolectada por varias vías y mostrarla de manera inteligente y entendible. Esta capacidad es implementada vía Servicio de Reportes llevando así datos útiles a las personas que deberán tomar decisiones en el proyecto.

d) Portal de Equipo

Provee además un repositorio de información útil para todo el equipo de desarrollo, donde se puede mostrar desde información de la metodología como también información relevante sobre el estatus del proyecto, guías sobre la realización de actividades, entre otras cosas

2.6.2. Visual Studio 2005 para Arquitectos de Software.

Este componente tiene la capacidad de análisis y visión que se puede tener al crear y analizar los distintos diagramas que provee. Esta especialmente pensado para sofocar los problemas que se tienen en la implementación de la arquitectura y despliegue de la solución, mostrando de manera simple los posibles conflictos que se pueden tener en ambas etapas.

Puede escribir un pequeño programa lanzándose directamente a los códigos, pero para planearlo, comunicarlo y documentarlo para conseguir la aprobación. Los cuatro objetivos del diseño visual son, por lo tanto, los siguientes:

- Ayudarle a visualizar el sistema que desea.
- Permitirle especificar la estructura o conducta de un sistema.

- Proporcionarle una plantilla que le guíe al construir un sistema.
- Documentar las decisiones que realice.

Tradicionalmente, los procesos de diseño como el Proceso Racional Unificado han tratado el diseño y la programación como disciplinas separadas, al menos en términos de soporte de herramientas. Se utiliza una herramienta de modelado visual para diseñar un IDE²⁵ diferente para codificar. En el caso del desarrollo de software, conlleva en realidad más costes cambiar una línea de código que un diagrama de diseño. Por estas razones, ha habido una tendencia hacia herramientas que permiten dos representaciones y, de este modo, tratando en lo fundamental el diseño y la codificación como dos puntos de vista de la misma actividad. Si la vista de diseño y la vista de código son representaciones alternativas pero equivalentes.

Es claro que si se utiliza el modelo visual como una herramienta de comunicación, la notación debe entenderse de forma común. En términos de UML esto significa un estándar de la industria, una notación de propósito general que se refiere a todas las cosas para todo el mundo. En términos de Visual Studio 2005, una serie de notaciones de dominio específico altamente preparadas para transmitir la información para la cual se diseñaron.

2.6.2.1. ESTRATEGIA DE MODELADO DE MICROSOFT.

La estrategia de modelado de Visual Studio 2005 Team System de Microsoft se basa en tres ideas clave:

- a) Desarrollo controlado por modelos (MDD)
- b) Lenguajes de dominio específico (DSL),
- c) Fábricas de software.

²⁵ IDE. (Entorno integrado de desarrollo) Aplicación compuesta por un conjunto de herramientas útiles para un programador.

a) Desarrollo controlado por modelos (MDD)

MDD consiste en sacar el máximo valor al esfuerzo de modelado mediante la obtención de tanta información como sea posible de los distintos modelos para la implementación. Esto es con el fin de cambiar la forma en que los desarrolladores perciben el valor del modelado, cambiando la percepción en que el modelado es una actividad marginalmente útil que precede al desarrollo real.

El Desarrollo controlado por modelos se traduce en la funcionalidad de Visual Studio 2005:

- Validación automatizada de la configuración de la aplicación con respecto a las restricciones del entorno de alojamiento.
- Generación de envolturas de servicios Web para salvar las distintas tecnologías de implementación.
- Sincronización automática entre el código fuente y los modelos de clase (por tanto, no hay problemas con la generación de código ni con las fases de ingeniería inversa)
- Producción de informes de implementación para ayudar a la implementación automatizada a través de herramientas de generación de código externas.

Los modelos se consideran artefactos de desarrollo de primera clase, los desarrolladores escriben un código menos convencional y el desarrollo es, por tanto, más productivo y ágil. Además, fomenta la idea entre todos los participantes, desarrolladores, diseñadores, analistas, arquitectos y el personal de operaciones, de que el modelado, realmente, añade valor a sus esfuerzos.

b) Lenguajes de dominio específico

Un lenguaje de dominio específico (DSL) es un lenguaje de modelado que satisface ciertos criterios. Por ejemplo, un lenguaje de modelado que desarrolle servicios Web debe contener conceptos como métodos y protocolos. El lenguaje de modelado debe, además, emplear nombres

significativos para los conceptos, campos y métodos en lugar de atributos y operaciones. Los nombres deben obtenerse del vocabulario normal del dominio.

La idea de los lenguajes DSL²⁶ no es nueva y puede que ya haya utilizado alguno para manipular bases de datos (lo que se conoce como SQL²⁷) o una definición de esquemas XML²⁸ (lo que se conoce como XSD²⁹).

Visual Studio 2005 abarca esta idea proporcionando lenguajes de dominio específico para tareas específicas. Los lenguajes de dominio específico permiten utilizar modelos visuales no sólo para crear documentación de diseño, sino también para capturar información de una forma determinada, de manera que pueda procesarse con facilidad, aumentando la posibilidad de compilar modelos en códigos.

c) Fábricas de software

La iniciativa de las fábricas de software consiste en unir las ideas del desarrollo controlado por modelos y del lenguaje de dominio específico para apoyar lo siguiente:

- Generación complementa o parcial de artefactos, como código fuente y archivos de configuración, a partir de otros artefactos, especialmente modelos.
- Sincronización de artefactos relacionados, como el modelo y el código, durante el desarrollo y validación de artefactos construidos manualmente.
- La aplicación de patrones y de las mejores prácticas de la industria controladas por orientación con contexto.

²⁶ DSL. Lenguaje de Dominio Especifico.

²⁷ SQL. Lenguaje de consulta estructurado.

²⁸ XML. lenguaje de marcas extensible.

²⁹ XSD. Un XSD o XML Schema, es un vocabulario basado en XML para describir instancias de documentos XML.

La clave de todo esto se encuentra en un Esquema de Fábrica de Software, que relaciona el trabajo hecho en un nivel de abstracción, en una parte del sistema o en una fase del ciclo de vida, con el trabajo hecho en otros niveles o en otras partes y fases.

Vemos las fábricas como la base de un ecosistema amplio en el que participan nuestros clientes y socios, creando fábricas personalizadas en las bases que suministramos y ofreciendo componentes de fábrica a otros miembros del ecosistema.

2.6.2.2. DISEÑADORES VISUALES

La serie de diseñadores visuales, proporciona herramientas para modelar el aspecto estático de su arquitectura de aplicación, infraestructura de implementación y diseño de objetos de nivel más bajo. La siguiente, es la lista completa de los nuevos diseñadores que trataremos:

- a) Diseñador de aplicaciones.
- b) Diseñador de centros de datos lógicos.
- c) Diseñador de sistemas.
- d) Diseñador de implementación.
- e) Diseñador de clases.

a) Diseñador de Aplicaciones

Permite la creación de diagramas basados en los componentes de la aplicación, desde que tipo son hasta los requerimientos específicos de dichos componentes. Sus principales características son la definición de interfaces de servicio de manera ágil y la transformación del modelo a un conjunto de proyectos de Visual Studio de manera rápida lo que tiene por objetivo reducir los tiempos en la implementación de dicha Arquitectura.

b) Diseñador de Centros Datos Lógicos

Permite definir de manera manual o automática la infraestructura con la que se cuenta en producción, sus configuraciones y sus características de manera tal que en pasos posteriores se pueda definir si los requerimientos de la aplicación son satisfechos por dicha infraestructura

c) Diseñador de Sistemas

De manera sencilla nos permite crear pequeños sub-sistemas que definirán componentes reutilizables y brindaran un análisis más detallado de los sistemas que conforman el modelo de Arquitectura

d) Diseñador de Implementaciones

Permite empalmar los modelos de Application Designer y Logical Datacenter Designer con el objetivo de definir como será realizado el deployment de la aplicación y además poder analizar los problemas de requerimientos que puedan presentarse.

e) Diseñador de clases

Visual Studio Team 2005 proporciona un diseñador de clases que se parece más a su homónimo en UML.

El diseñador de clases puede plantearse de dos formas:

- De código a diagrama.- Los diagramas de clase como una manera de representar código en imágenes
- De diagrama de clases a código.- Donde los diagramas de clases deben utilizarse para controlar el diseño de alto nivel y la estructura general del código

El diseñador de clases no está sólo en Visual Studio 2005 Team Edition para Arquitectos de Software. También puede realizar modelado de clases en la edición para desarrolladores y en la edición para probadores.

2.6.3. VISUAL STUDIO 2005 PARA DESARROLADORES DE SOFTWARE.

Visual Studio es una herramienta excelente para escribir código. Mejorada al aportar al programador un mayor número de herramientas para la escritura de código de mejor calidad.

Provee la funcionalidad necesaria para la realización de diversas tareas como ampliar las capacidades de generación de código, el análisis estático de código o pruebas unitarias, estas herramientas son fundamentales para poder así prever diversos problemas que pudieran presentarse en Producción.

Brinda la capacidad de estipular diversas políticas para el Check-In³⁰ asegurando además de que el código que se ingrese al Source Control sea de calidad. A través de otras características como cobertura de código que permite al desarrollador de software detectar diversos riesgos que pueden ir desde problemas de seguridad y desempeño^[20].

a) Pruebas Unitarias

Permite la realización de pruebas unitarias sobre el código, para determinar si los componentes de manera individual funcionan como han sido diseñados. También permite realizar Test Driven Development con lo cual se puede reducir en gran medida los defectos que serán encontrados posteriormente.

La integración de pruebas unitarias y la estrecha integración con un marco de trabajo de pruebas facilitan la creación de código de mejor calidad en una fase anterior del ciclo de vida reduciendo los costos e incrementando la productividad.

³⁰ Check-In (llegada). Crea un conjunto de cambios en el servidor Control de código fuente

^[20] <http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnvs05/html/utfwvs05tmsys.asp>

b) Cobertura de Código.

Una característica muy importante ya que permite que el desarrollador de Software pueda determinar qué porcentaje del código ha sido probado a través de Pruebas Unitarias, e incluso a un detalle tal de saber qué línea o líneas de código no se les ha realizado ninguna prueba o a sido una prueba parcial.

c) Análisis estático

Provee la capacidad de analizar el código contra reglas preestablecidas para determinar deficiencias que pueden ir desde la nomenclatura de los objetos hasta problemas en seguridad.

El análisis estático proporciona soporte tanto para código gestionado como para código nativo; el perfilado de código incorpora la visualización en vistas secuenciales, ubicación de objetos y vistas de funciones, además de un verificador de aplicaciones. Esto significa que los desarrolladores entenderán mejor las consecuencias del trabajo previo y tendrán una visión de la coherencia global del código generado conjuntamente con las especificaciones.

2.6.4. VISUAL STUDIO 2005 PARA TESTERS DE SOFTWARE.

Proporcionar un conjunto de herramientas de prueba integradas por parte de Microsoft. Facilita herramientas avanzadas para la realización de pruebas unitarias, pruebas de carga, web y manuales que permitirán al Ingeniero de Pruebas determinar si la solución cumple o no con los requerimientos de calidad, estas pruebas además de contar con asistentes funcionales, muestra los resultados de manera clara brindando una gran facilidad en el análisis de los mismos.

a) Las pruebas unitarias y la cobertura de código.

Las pruebas unitarias y la cobertura de código son capacidades que extienden tanto Team Edition para Probadores de Software como Team Edition para Desarrolladores de Software. Esto es importante porque permite a los desarrolladores y probadores realizar pruebas tempranas, frecuentes e iterativas con las pruebas unitarias. La cobertura de código permite a los profesionales de ambas áreas conocer qué partes de la aplicación han sido probadas y cuales permanecen sin probar. Esto también facilita la priorización de código y los recursos de pruebas, ya que ciertas áreas de código tienen más riesgo de fallo o son más importantes para el negocio, de modo que deberíamos dirigir ciertos recursos a esas áreas en primer lugar.

b) Pruebas de Carga

Realiza pruebas simulando múltiples usuarios concurrentes, para determinar el comportamiento de la solución cuando esta sea implementada en producción, permitiendo así una previsión de los posibles problemas generalmente relacionados al funcionamiento.

c) Pruebas Web

Componente que permite realizar pruebas relacionadas al flujo de una aplicación Web como también la validación de los datos mediante reglas, permitiendo así probar una interacción real de usuarios con la misma. Las pruebas de carga abordan el rendimiento en el contexto de servicios Web y páginas Web, mediante un script³¹, patrones de carga y un contador de rendimiento para recopilar y proporcionar monitorización de umbral.

³¹ Script. Guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades.

d) Manual Test

En muchas ocasiones ciertas pruebas no pueden ser automatizadas. Sin embargo esta herramienta provee la capacidad de integrarlas al conjunto de pruebas, como también reportar los defectos encontrados a través de elementos de trabajo.

2.7. METODOLOGÍA UTILIZADA

MVSTS incorpora las instancias Microsoft Solución Framework de procesos predeterminados que se encuentra dentro del portal del equipo y que pueden llegar a ser personalizados por miembros de empresas de desarrollo.

2.7.1. MICROSOFT SOLUTION FRAMEWORK

Es un conjunto probado de las mejores y más importantes prácticas para ayudar a guiar los proyectos de desarrollo de software. MSF se integra con Team System y se manifiesta a sí mismo como una guía de procesos dentro del portal del equipo y a través de la automatización mediante elementos de trabajo y productos de trabajo. Además se puede personalizar y llevar a cualquier entorno de desarrollo^[21].

Brinda un soporte computacional para el control y seguimiento de metodologías de desarrollo. VSTS viene configurado con dos metodologías MSF: MSF for Agile Software Development y MSF for CMMI Process Improvement.

2.7.2. MSF FOR AGILE SOFTWARE DEVELOPMENT

Está orientada a desarrollos ágiles y es el primer proceso ágil que considera el ciclo completo de desarrollo y a todos los roles del equipo. Presenta un

^[21] Profesional Visual Studio 2005 Team System de Jean-Luc, Tony Loton, Erick Gunvaldson, Christopher Bowen, Noah Coad, Darren Jefford. Pág. 653.

esquema iterativo e incremental, basado en escenarios, más que en casos de uso y está orientada para equipos pequeños o equipos grandes subdivididos en equipos pequeños^[22].

La plantilla MSF for Agile Software Development tiene un diseño óptimo para proyectos pequeños con un calendario de entrega rápido. Puede ser conveniente elegir la plantilla MSF for Agile Software Development si su empresa:

- No tiene muchos procesos documentados y no está interesada en desarrollar procesos formales.
- Tiene equipos de desarrollo de software reducidos.
- Puede dar cabida a varios equipos de desarrollo pequeños trabajando de varias maneras.
- Tiene ciclos de desarrollo de software cortos (medidos en semanas o meses).
- Admite varias versiones de software en un año natural^[23].

2.7.3. MSF FOR CMMI PROCESS IMPROVEMENT

MSF for CMMI Process Improvements es una versión más clásica de MSF orientada a proyectos con mayor procedimiento y es el primer proceso ágil basado en CMMI que ayuda a las organizaciones a operar a un nivel de CMMI. Si bien está basado en el modelo de MSF 3.0 presenta WorkItems Adicionales y un extensivo reporte.

El modelo de proceso de MSF for CMMI Process Improvement se ha creado en el entorno empresarial, en gran medida alejado del ámbito de desarrollo de software, por profesionales convencidos de que la formalización y mejora de las prácticas más apropiadas mediante la aplicación de las conclusiones

^[22] <http://arielschwindt.spaces.live.com/blog/cns!61CF6029461DCD5E!497.entry>.

^[23] [http://msdn2.microsoft.com/es-es/library/ms400752\(VS.80\).aspx](http://msdn2.microsoft.com/es-es/library/ms400752(VS.80).aspx).

aprendidas es vital para lograr el éxito a largo plazo. Puede ser conveniente elegir la plantilla MSF for CMMI Process Improvement si su empresa:

- Desea evaluar las prácticas empresariales actuales.
- Tiene grandes equipos de desarrollo de software.
- Necesita integrar grupos u organizaciones tradicionalmente independientes.
- Puede proporcionar orientación para los procesos de la calidad.
- Tiene ciclos de desarrollo de software más prolongados^[24].

Cada una de estas metodologías se define como un conjunto de archivos XML susceptibles de ser modificados, extendidos y soportan los siguientes elementos:

- Tipos Elementos de Trabajo, ciclos de trabajo
- Políticas de Check In
- Plantillas de Documentos
- Reportes
- Permisos y Grupos
- Ayuda Integrada
- Criterios de éxito por fase

Al poseer un esquema extensible, nos permite definir nuestra propia Metodología. La manera más rápida de crear nuestra propia metodología es editar el template de alguna de las existentes, que es un conjunto de archivos XML, realizando los cambios que considere necesarios definiendo por ejemplo nuevas tareas, fases y entregables y luego importando esta nueva metodología a VSTS.

^[24] <http://www.microsoft.com/spanish/msdn/latam/alm/>

MSF for Agile Software Development.	MSF for CMMI Process Improvement.
Habilita la agilidad del proceso de desarrollo de software.	Extiende MSF for Agile Software Development con soporte para auditar, comprobación y procesos formales.
Las personas son más importantes que el proceso.	Se centra en la organización.
Mejoras a partir del postmortem de iteración y otras revisiones informales.	Mejoras continuas formales y garantías de calidad.
Se basa en cualidades y habilidades de los desarrolladores.	Se basa en el proceso y en el cumplimiento del proceso, no en las especificaciones.
Se basa en el Manifiesto de Agile.	Se basa en los trabajos de W. Edgard Deming y Philip Crosby.
Centrado en el proyecto.	Centrada en la Organización.
Fuente[25]	
Tabla 2.1 Elegir el Proceso adecuado.	

2.8. CONSIDERACIONES

Las exigencias actuales de calidad en el desarrollo de Software han aumentado considerablemente, lo que ha llevado a la necesidad de implementar disciplinadamente metodologías y como consecuencia la creación de herramientas como MVSTS para integrar dentro de las herramientas mecanismos que permitan facilitar la realización de cada uno de esos procesos sin importar el rol que cada miembro del equipo desempeñe.

Los principales beneficios de Microsoft Visual Studio Team System son la mayor adaptabilidad a los negocios y la colaboración con los responsables de los departamentos de informática, a través de un desarrollo de software que responde dentro del modelo familiar de las herramientas de Microsoft, supondrá importantes ahorros de costes gracias al incremento de la productividad y la comunicación a través de una información integrada, una captura de datos

[25] Profesional Visual Studio 2005 Team System de Jean-Luc, Tony Loton, Erick Gunvaldson, Christopher Bowen, Noah Coad, Darren Jefford. Pág.666 - 667.

automatizada por el conjunto de herramientas del ciclo de vida, un portal común para proyectos, consultas e informes cruzados entre herramientas, y servicios de enlace comunes y de notificación. Con el tiempo, MVSTS evolucionará más allá de su objetivo actual para abarcar una integración con las herramientas de planificación y gestión de proyectos, un mayor soporte para la estimación de proyectos y su gestión, y enlaces con la gestión de sistemas para un mejor soporte de las operaciones.

Para más información se recomienda visitar el siguiente enlace <http://msdn2.microsoft.com/es-es/>