Universidad de las Ciencias Informáticas
Facultad 5

Título: Adaptación de *Microsoft Solutions Framework Agile* (MSF Ágil) al proceso de desarrollo de videojuegos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Yalina Arias Nápoles

Yulien Martínez Conde

Tutor: Ing. Igr Alexander Fernández Saúco

Co-tutor: Ing. Reina Mercedes Bauta Gómez

Ciudad de la Habana, Junio del 200

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Yalina Arias Nápoles

_____

Yulien Martínez Conde

_____

Tutor: Ing. Igr Alexander Fernández Saúco

_____

Co-Tutor: Ing. Reina Mercedes Bauta Gómez

_____

DATOS DE CONTACTO

Nombre y Apellidos: Igr Alexánder Fernández Saúco
Edad: 28
Ciudadanía: Cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero Informático
Categoría Docente: Profesor Instructor
E-mail: alexanderfs@uci.cu
Jefe del grupo de Arquitectura y Tecnologías, Dirección Técnica, IP – UCI

Nombre y Apellidos: Reina Mercedes Bauta Gómez
Edad: 24
Ciudadanía: Cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniera en Ciencias Informáticas
Categoría Docente: Profesora Adiestrada
E-mail: rbauta@uci.cu

DEDICATORIA

A mis padres que son la razón de mi ser.

A mi hermanita Yamy, que comparte su vida conmigo.

A mis amigos que hacen que cada día sea una vida.

Y al amor que le da sentido a mis sueños.

Yulien Martínez Conde

A mis padres, por su infinito amor.

A mi familia, en especial a mi tía Ary por su cariño y apoyo incondicional durante estos cinco años.

Y a mis amigos, por darme fuerzas cada día para alcanzar este sueño.

Yalina Arias Nápoles

AGRADECIMIENTOS

A la Uci y a Fidel y por permitirme realizar mi sueño.

A la Virgen y a las personas que más amo en este mundo: mi familia. En especial a mi madre, que ha sido y será mi fiel amiga, a mi padre por apoyarme siempre y creer en mí, a mi hermanita Yamy por ser tan linda conmigo, a mi tía Tania por escucharme siempre, a mis abuelos y a mis primas Taty y Katia por tenerme presente.

A mis amigos y hermanos de estos cinco años por ser para toda la vida y formar parte de mí: A Yilver e Isabel por su cariño incondicional y su amistad sincera, a Yaly, amiga y compañera de tesis por tener tanta paciencia conmigo, a Eli y a Yazmín por ayudarme a ser mejor persona, a Oda y a Brenda por incluirme entre sus amistades, a Lianyi por cuidar de mí y a Yirka y a Lenna por ser mis guías.

A mis compañeros del proyecto Juegos Consola por contribuir con mi superación. A mis tutores Igr y Reina por su apoyo y a todos aquellos que de alguna manera han estado al tanto de mí.

Yulien Martínez Conde

A mi mamá que es la persona que más amo en este mundo. A mi papá por su inmenso amor. A mi tía del alma por su apoyo incondicional. A toda mi familia, a mi hermana, mis tías y primas. A mi abuelo adorable que ha sido otro padre para mí y a mi abuela Josefina, que aunque no esté entre nosotros, este también fue su sueño. A Lester, por su amor durante estos cinco años, y a su familia que ha llegado a ser también mía, en especial a Oscar, Lupe, Maritza y Grisel por el cariño que me han brindado.

A mis amigos, en especial a Dayi y a Yuly, que han sido personas excepcionales y además Yuly mi compañera de tesis, a ella gracias por su amistad y confianza; a Yilver, Isa y Eli que me han permitido ser parte de esa gran amistad que los une. A Yirki y Lenna por su apoyo y confianza.

A mis tutores Igr y Reina, por su ayuda en la realización de este trabajo, y a mis compañeros del proyecto Juegos Consola.

A la Revolución, a Fidel, y a todos, gracias por permitir que se haga realidad mi sueño.

Yalina Arias Nápoles

RESUMEN

El presente trabajo de diploma tiene como propósito la adaptación de una estrategia ágil para el desarrollo de *software* específicamente para el desarrollo de videojuegos, en el proyecto Juegos Consola, su título es “Adaptación de *Microsoft Solutions Framework Agile* (MSF Ágil) al proceso de desarrollo de videojuegos.”

Surge de la necesidad de poner en manos del grupo de desarrollo de Juegos Consola de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), una estrategia o marco de trabajo que agilizara el proceso de desarrollo de videojuegos, su aplicación tendrá como objetivo aplicaciones de pequeña escala y riesgo limitado.

MSF Ágil ofrece una guía sobre cómo organizar personas y proyectos, contiene las mejores prácticas de la administración de proyectos para planificar, desarrollar y desplegar soluciones tecnológicas de *software,* de una forma orientada a los cambios que el cliente requiera para el producto en cualquier etapa dentro del ciclo de vida del proyecto.

La investigación estuvo dirigida fundamentalmente al análisis de MSF Ágil, a la interactividad entre *Visual Studio Team System* (VSTS) y *Visual Studio Team Foundation Server* (TFS) a través de los procesos de MSF Ágil y se describe además el proceso de gestión de proyectos guiado por la estrategia. Una de las principales tareas llevadas a cabo durante la investigación fue la comparación del proceso de desarrollo de *software* actual según el expediente del proyecto, con el mismo proceso guiado por MSF Ágil. Y el principal resultado fue la personalización de la plantilla de procesos MSF para el desarrollo ágil, dando lugar a la nueva plantilla de procesos para proyecto Juegos Consola: **Videojuegos UCI MSF para el desarrollo ágil.**

PALABRAS CLAVE

Palabras claves: investigación, videojuego, realidad virtual, metodología, ágil, proceso, desarrollo, *software*, herramientas, integración, MSF Ágil, proyecto, plantilla.

ÍNDICE

INTRODUCCIÓN

La informática abarca un conjunto de conocimientos y herramientas científicas, técnicas y tecnológicas que, para enmarcar su utilidad, se podría decir que se encarga del tratamiento racional y estructurado de la información por medios automáticos electrónicos digitales. En la informática convergen los fundamentos de las ciencias de la computación, la programación y las metodologías para el proceso de desarrollo de software, la arquitectura de computadores, las redes de datos como Internet, la inteligencia artificial, así como determinados temas de electrónica. Se puede entender por informática a la unión sinérgica de todo este conjunto de disciplinas. En pocas palabras se puede decir que para el mundo de hoy, la informática está siendo el motor impulsor de la más profunda revolución tecnológica conocida, transformando la vida social en lo que se ha dado por llamar: Sociedad de la Información.

Esta revolución tecnológica se debe fundamentalmente a los avances alcanzados en la industria del *software* ya que es posible convertir ideas en productos con relativamente pocos recursos, si se compara con otras industrias. La industria del *software* y los servicios informáticos han pegado un gran salto en la calidad de los productos, que se ve reflejado en el creciente interés de empresas importantes de la mayoría de los países por adquirir tecnología nacional, y que se materializa con el perfeccionamiento creciente de los procesos de desarrollo de *software* para la producción.

Un proceso de desarrollo de *software* tiene como propósito la producción eficaz y eficiente de un producto *software* que cumpla las expectativas del cliente especificadas en requisitos. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas.

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos *software*.

Un producto *software* en sí es complejo, es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea. Es intangible y por lo general muy abstracto, esto dificulta la definición del producto y sus requisitos, lo que provoca que estos últimos sean difíciles de consolidar

tempranamente. De esta forma, los cambios en los requisitos son inevitables, no sólo después de entregado en producto sino también durante el proceso de desarrollo.

El proceso de desarrollo de *software* no es único. No existe un proceso de desarrollo de *software* universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de *software*.

Sobre la naturaleza caótica del *software* se encontró una importante referencia en el artículo (Brooks, 1987) donde se analizan las características intrínsecas del *software*. En el mismo, el autor tomaba la realidad de la disciplina mencionando los grandes problemas que planteaba el desarrollo de *software*. Motivo por el cual surge la necesidad de la estandarización, materializada con la aparición de modelos de desarrollo de *software*.

En (Sommerville, 2002) se define modelo de proceso de desarrollo de *software* como “Una representación simplificada de un proceso de *software*, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del *software* es una abstracción de un proceso real.”

Estos modelos no son descripciones definitivas de procesos de *software*; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de *software*. La utilización de estos implicó un conjunto de deficiencias en el proceso de desarrollo de *software*, ya sea, porque el *software* bien diseñado, no se ajustaba a las necesidades del usuario, por lo que era rechazado; las iteraciones eran muy costosas e implicaban rehacer trabajo debido a la producción y aprobación de documentos; por la inflexibilidad en la evolución al incorporar nuevos requisitos; por dificultades de respuesta a cambios en los requisitos, además en ocasiones se necesitaban para el rápido desarrollo, herramientas que podían ser incompatibles con otras o que poca gente sabía utilizar.

El hecho de contar con potentes y eficaces notaciones y herramientas no significa que éstas se utilicen de la forma más eficiente posible. Se necesita entonces de un procedimiento que especifique cómo aplicarlas si se quiere obtener un producto satisfactorio en relación con la utilización de ambos métodos. Este hecho ha traído consigo un gran interés en las metodologías de desarrollo las cuales se han iniciado con un gran auge (Letelier, et al., 2006). Las metodologías se basan en una combinación de los modelos de proceso

(cascada, evolutivo, incremental, entre otros), con el objetivo de estructurar el proceso de desarrollo utilizándolas como un marco referencial para lograr una forma ordenada y predecible de trabajo en la cuál los resultados se podrán garantizar. En un proceso de desarrollo de *software*, la metodología define QUIÉN debe hacer QUÉ, CUÁNDO y CÓMO; por esa razón no existe una metodología de desarrollo de *software* universal, dado que las características de cada proyecto (equipo de desarrollo, recursos, entre otros) exigen que el proceso sea configurable.

Mayormente se realiza el diseño de *software* de manera "rígida" según las especificaciones descritas por el cliente, de forma tal que si en la etapa final o de prueba, el cliente solicita un cambio, se dificulta esta modificación por los imprevistos que implica. Esto constituye uno de los factores que ocasiona un atraso en el proyecto, la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido.

Los procesos de desarrollo de *software* están mayormente guiados por metodologías tradicionales donde muchas veces debido a su gran planificación y rigidez resulta muy engorrosa su aplicación en proyectos donde se necesita obtener rápidamente un producto. Para resolver esta situación se presentan las metodologías ágiles con una forma más flexible de desarrollo.

Las metodologías ágiles están revolucionando la manera de producir *software*. Muchas de las empresas productoras de *software* están encaminando su proceso de *software* bajo estas metodologías debido a las facilidades que presentan, éstas constituyen una solución a medida para muchos proyectos de *software*, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

En (Bauta, et al., 2007) se plantea que los informáticos en el afán de automatizar otros procesos han descuidado sus propias tareas, ha dado lugar a un sinnúmero de errores que han provocado el fracaso parcial o total de algunos proyectos. La poca motivación, las fricciones entre los desarrolladores y el cliente, las expectativas poco realistas, la falta de un *manager* efectivo para el proyecto, la pérdida de tiempo, el diseño no adecuado, la falta de supervisión y de estimaciones, los requerimientos excesivos, la sobre-estimación de las ventajas del uso de una nueva herramienta, el cambio de herramientas a mitad del proyecto y la falta de un control de código fuente automático son, por citar algunos, los errores más comunes relacionados con los procesos de desarrollo, junto a los productos y las tecnologías que se

emplean en la elaboración de un *software*. Es entonces cuando resulta muy común preguntarse qué metodología utilizar. Cada proceso lleva en sí una forma lógica a seguir si se quieren obtener los resultados que se esperan.

La no aplicación de alguna metodología y herramientas en el proceso de producción de *software* de realidad virtual, presupone una mala gestión del mismo, según se expone en (Bauta, et al., 2007). Lo anteriormente dicho se corresponde con la situación existente en el proyecto Juegos Consola perteneciente a la Facultad 5 de la UCI, donde no se ha aplicado, en el proceso de desarrollo de los videojuegos, una metodología o buenas prácticas que mejoren de forma significativa el proceso de producción de este tipo de *software*, lo que trae como consecuencia, documentación excesiva dada las características específicas del proyecto, poca adaptabilidad ante los cambios de requisitos, un proceso más costoso en tiempo y recursos, mala organización y comunicación del equipo, ineficiente asignación de tareas, escasas pruebas, deficiencias en el uso de métricas, retraso en la culminación y como consecuencia de todo lo anterior, demora en la entrega del producto final.

Según la propuesta realizada en (Bauta, et al., 2007) para el proceso de producción de videojuegos en el proyecto Juegos Consola, de adoptar *Microsoft Solutions Framework Agile* (MSF Ágil) como mejor estrategia para disminuir los problemas del mismo, se propone como **problema científico** lo siguiente:

¿Cómo mejorar el proceso de desarrollo de videojuegos en el proyecto Juegos Consola perteneciente a la Facultad 5 de la UCI?

Por lo anteriormente expresado el **objeto de estudio** lo constituye el proceso de desarrollo de *software* y el **campo de acción** se centra en el proceso de desarrollo de videojuegos en el proyecto Juegos Consola perteneciente a la Facultad 5 de la UCI.

Se definió para la realización de la investigación como **objetivo general**, adaptar MSF Ágil al proceso de desarrollo de videojuegos.

Los **objetivos específicos** que dan cumplimiento al objetivo general son los siguientes:

- Analizar MSF Ágil, así como la interactividad entre *Visual Studio Team System* (VSTS) y *Visual Studio Team Foundation Server* (TFS) a través de los procesos de MSF Ágil.

- Describir el proceso de gestión de proyectos usando MSF Ágil.
- Comparar el proceso de desarrollo de *software* actual con un proceso guiado por MSF Ágil.
- Personalizar la plantilla de procesos MSF para el desarrollo ágil del TFS.

Como **hipótesis** se plantea que con la adaptación de MSF Ágil al proceso de desarrollo de videojuegos en el proyecto Juegos Consola perteneciente a la Facultad 5 de la UCI, entonces mejorará considerablemente dicho proceso.

Para guiar la investigación científica se utilizarán los **métodos**:

Teóricos

- Analítico _ Sintético: Se analizan las teorías, documentos entre otros, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
- Inductivo _ deductivo: Son formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular a lo general, como desde el análisis de los elementos generalizadores a uno de menor nivel de generalización.
- Análisis histórico lógico: El método lógico para poder descubrir las leyes fundamentales de los fenómenos, debe basarse en los datos que le proporciona el método histórico, de manera que no constituya un simple razonamiento especulativo. De igual forma el método lógico debe descubrir las leyes, la lógica objetiva del desarrollo histórico del fenómeno y no limitarse a la simple descripción de los hechos.

Empíricos

- Observación: Registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los hechos y acontecimientos pertinentes de acuerdo con algún esquema previsto.

Los posibles resultados a obtener son:

- Personalización de la plantilla de proceso MSF para el desarrollo ágil integrada al TFS en el proyecto Juegos Consola.
- Uso de herramientas integradas en el proceso de producción de *software* que permitan la gestión del mismo.

Este documento se encuentra estructurado en tres capítulos. El primer capítulo contiene el marco teórico referencial de la solución técnica sobre la adaptación de MSF Ágil al proceso de desarrollo de videojuegos y el uso de las herramientas integradas al proceso de producción de *software*: *Visual Studio Team System* y *Visual Studio Team Foundation Server*. En el segundo capítulo se establece una comparación entre el proceso de desarrollo de *software* actual en el proyecto y uno guiado por MSF Ágil haciendo uso de las herramientas integradas. En el tercer capítulo, se describe el proceso de personalización de la plantilla de MSF Ágil para el TFS.

Nota aclaratoria importante: Casi la totalidad de la bibliografía que se tuvo en cuenta para realizar la presente investigación fueron textos e imágenes en idioma inglés, dado que la tecnología que se aplicará es creación de la corporación *Microsoft*. Como el idioma en que se defiende este trabajo de diploma es el español, se aclara que los términos que aparecen escritos en inglés no tienen una traducción literal al lenguaje que compete a estos textos y por eso su aparición se conserva en el lenguaje original.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se hace un análisis de MSF Ágil, así como la interactividad entre *Visual Studio Team System* y *Visual Studio Team Foundation Server* a través de los procesos de MSF Ágil, además se argumenta el uso de estas herramientas como apoyo a la solución técnica, adaptación de MSF Ágil al proceso de desarrollo de videojuegos, por las grandes ventajas que ofrece y que hasta el momento las hacen irremplazables a pesar de formar parte del conjunto de soluciones de *Microsoft*. Se identifican las ventajas de adaptar MSF Ágil al proceso de desarrollo de videojuegos en el proyecto Juegos Consola y se describe el proceso de gestión de proyectos usando MSF Ágil.

Todo desarrollo de *software* implica riesgos y complejidad en la gestión del mismo, por tanto, si no se sigue una metodología adaptable o se aplican prácticas que se ajusten a las condiciones del equipo/proyecto, con certeza los clientes y desarrolladores quedarán insatisfechos con el resultado.

Existen técnicas innovadoras a las barreras comunes del proceso de desarrollo que proporcionan alternativas a usar en un proceso ágil. De allí que uno de los primeros pasos del equipo de desarrollo ante la disyuntiva de cuál metodología usar, debe ser analizar si el proyecto se adapta al enfoque ágil.

Microsoft Solutions Framework (MSF) cabe en el marco de las soluciones de *Microsoft* como un sistema integrado de la dirección de procesos que incluye una estrategia ágil y modelos formales y proporciona un espacio para poner una solución en ejecución, modificada para requisitos particulares de una amplia variedad de proyectos. No puede haber un único proceso de desarrollo de *software* para todos los proyectos de desarrollo de *software*. Ésta es la filosofía central detrás del marco de las soluciones de *Microsoft*.

MSF para el desarrollo ágil del *software* es un proceso que utiliza muchas de las ideas incorporadas al VSTS. El proceso MSF incorpora prácticas probadas desarrolladas en *Microsoft* alrededor de requisitos, de diseño, de seguridad, de funcionamiento, y de la prueba. El VSTS es una plataforma productiva, integrada, y herramientas extensibles del ciclo de vida del desarrollo del *software* que ayuda a los equipos

de trabajo a ir mejorando la comunicación y la colaboración a través del proceso de desarrollo del *software*.

Conceptos fundamentales:

Metodología

Desde el punto de vista informático, un proceso de software detallado y completo suele denominarse "Metodología" según plantea (Letelier, 2006) Adicionalmente una metodología debe definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, entre otros.

Proceso

Un proceso es una acción o sucesión de acciones continuas regulares, que ocurren o se llevan a cabo de una forma definida, y que llevan al cumplimiento de algún resultado; una operación continua o una serie de operaciones (Peppard, et al., 1996).

Un proceso define *quién* está haciendo *qué*, *cuándo*, y *cómo* alcanzar un determinado objetivo. En la ingeniería del *software* la idea es construir un producto *software* o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de *software* de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible. El efecto global es fomento de una visión y una cultura comunes, según (Jacobson, et al., 1999).

Software

Probablemente la definición más formal de *software* es la atribuida a la IEEE en su estándar 729: "La suma total de los programas de cómputo, procedimientos, reglas documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo" (IEEE, 1993). Bajo esta definición, el concepto de *software* va más allá de los programas de cómputo en sus distintas formas: código fuente, binario o ejecutable, además de su documentación: es decir, todo lo intangible.

Proceso de desarrollo de *software*

Un proceso de desarrollo de *software* es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema *software* (Jacobson, et al., 1999).

Metodología de Desarrollo de *Software*

"Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas informáticos". Definió (Maddison, 1983).

Ingeniería de *Software*

El (IEEE, 1993) ha desarrollado una definición más completa para ingeniería del *software* (Pressman, 1997):

"(1) La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del *software*; es decir, la aplicación de ingeniería al *software*.

(2) El estudio de enfoques en (1)".

En (Pressman, 1997) se caracteriza la Ingeniería de *Software* como "una tecnología multicapa".

Videojuegos

Nombre genérico con el que se conocen ciertos programas de carácter lúdico que pueden ser ejecutados en ordenadores o en otros dispositivos, también de base informática, llamados consolas. En los últimos años su desarrollo ha sido espectacular, tanto desde el punto de vista comercial como en lo que respecta a sus prestaciones y capacidades (Glosario.Net, 2006).

Iteración

Las iteraciones son siempre puntos de control, aprendizaje y planificación. El ciclo primario del planeamiento es la iteración (GUIDANCE, 2005).

1.1 *Microsoft Solutions Framework.*

Microsoft Solutions Framework es "un conjunto de principios, modelos, disciplinas, conceptos, lineamientos y prácticas probadas" elaborado por *Microsoft*. MSF se encuentra en estrecha comunión de principios con las metodologías ágiles, algunas de cuyas intuiciones y prácticas han sido anticipadas en las primeras versiones de su canon, hacia 1994.

Es un modelo muy utilizado para el desarrollo y gestión de proyectos, es importante mencionar que este modelo involucra dos modelos de desarrollo de *software* que son: el modelo cascada y el modelo espiral.

Aunque MSF también se presta como marco para un desarrollo fuertemente documentado y a escala de misión crítica que requiere niveles más altos de estructura, como el que se articula en CMMI (*Capability Maturity Model Integration*), PMBOK o ISO9000, sus disciplinas de ningún modo admiten la validez (o sustentan la vigencia) de modelos no iterativos o no incrementales.

En suma, muchos de los principios de las metodologías ágiles son consistentes no sólo con los marcos teóricos, sino con las prácticas de *Microsoft*, las cuales vienen empleando ideas adaptativas, desarrollos incrementales y metodologías de agilidad (algunas de ellas radicales) desde hace mucho tiempo.

A principios de los '90 *Microsoft* comenzó a recopilar a nivel interno las mejores prácticas en términos de procesos de desarrollo de *software*, no con la intención de establecer una metodología, sino con la idea de tener una colección de prácticas individuales de lo que funciona, aplicables dentro de determinados contextos.

Plantea (Spaces, 2005) que MSF no es como tal una metodología, sino prácticas que, ajustándose al contexto de proyecto (tamaño del equipo, frecuencia de entregas, entre otros) serán más o menos recomendables de aplicar. Se autodefine como "marco" y no como metodología, en oposición a una metodología prescriptiva, ya que parte de la base de que no hay una estructura de procesos óptima para las necesidades de todos los entornos de desarrollo posibles, es por ello que más que una metodología es un marco de desarrollo flexible.

Para el proyecto se podrán seleccionar aquellas prácticas que realmente agreguen valor al proceso. Cada práctica se compone de una secuencia de actividades tales como planear, construir y administrar, las mismas se describen en **ETVX** (**E**ntrada, **T**areas, **V**erificaciones, **V**alidaciones, E**x**it), un modelo de documentación de procesos para representar criterio de entrada/salida, tareas, verificaciones y validaciones.

Fig. 1.1 Actividades de MSF (Sánchez, 2004)

Microsoft ha recopilado las mejores prácticas empleadas por sus grupos de producto, organización interna de tecnología, sus clientes y socios de negocios en todo el mundo y ha reunido los factores de éxito que son comunes a todas ellas, integrándolos en modelos reutilizables con etapas y logros medibles que pueden guiar las decisiones tecnológicas en un contexto de negocios. Estos modelos conforman el ***Microsoft Solutions Framework***.

Como se plantea en (Bauta, et al., 2007) las prácticas generan resultados tangibles en forma de productos de trabajo. Estos resultados aunque análogos a los artefactos de RUP (*Rational Unified Process*), no son necesariamente los mismos. Todo proyecto se separa en cinco fases principales.

Fig. 1.2 Fases de MSF

Según (Consultores, 2006), las fases de MSF son las siguientes:

Visión y Alcance: En esta fase se definen los líderes del proyecto, se plantean cuáles son los objetivos que se persiguen con el trabajo y hasta dónde se quiere llegar con el proyecto, trata de unir el proyecto/producto a las experiencias de otros trabajos, el equipo debe saber lo que el cliente desea y cómo lo desea, además se realiza la evaluación inicial de riesgos del proyecto. La fase culmina con el hito Visión y Alcance aprobados.

Planificación: Es en esta fase cuando se termina la planificación del proyecto, el equipo prepara las especificaciones funcionales, se realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costo y cronogramas de los diferentes entregables del proyecto. Esta fase culmina con el hito Plan del proyecto aprobado.

Desarrollo: Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código), además se desarrolla la infraestructura del proyecto. La fase culmina con el hito Alcance completo.

Estabilización: En esta fase se conducen pruebas sobre la solución, que enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar los errores, solucionarlos y preparar la solución para el lanzamiento. La fase culmina con el hito *Release* listo aprobado.

Implantación: En esta fase se decide la tecnología base y sus componentes, estabiliza la instalación, traspasa el proyecto al personal soporte y operaciones, y obtiene la aprobación final del cliente. La fase termina con el hito Implantación completa.

Estos modelos de MSF tienen seis roles que corresponden a las metas de un proyecto y éstos son responsables de que las mismas sean cumplidas. Todos los roles de un proyecto tienen igual importancia en su aporte al mismo, mostrando que no es un modelo jerárquico, aunque éstos pueden tener diferentes niveles de actividades durante las etapas de desarrollo y ninguno puede ser omitido.

MSF cuenta además con modelos que se encargan de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso,

Modelo de Gestión de Riesgos, Modelo de Diseño de Soluciones, Modelo de Infraestructura, Modelo de Costo Total de Propiedad y finalmente el Modelo de Aplicación.

MSF incluye tres disciplinas: Disciplina de Administración de Proyecto, Administración de Riesgos y Administración de la Preparación, se centra en 2 de los modelos: Modelo de Equipo y de Proceso.

1.2 Enfoque ágil de MSF

La "Metodología Ágil" surge para acabar con el "burocratismo" de aquellas metodologías que como RUP necesitan mucha documentación. Es por eso que la "ágil" es más aceptada, el trabajador ve mejor el fruto de su desempeño y no se siente cansado en la medida que va evolucionando el proyecto; por eso se dice que esta estrategia da más valor al individuo y lo hace sentir más a gusto trabajando con ella.

La meta de los métodos ingenieriles es definir un proceso que funcionará bien con cualquiera que lo use. Los métodos ágiles afirman que ningún proceso podrá nunca "maquillar" las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo. Explícitamente puntualizan el trabajar a favor de la naturaleza humana en lugar de en su contra y enfatizan que el desarrollo de *software* debe ser una actividad agradable. (Fowler, 2003)

Lo expresado en el párrafo anterior apoya la propuesta realizada en (Bauta, et al., 2007) de aplicar una estrategia ágil en el proyecto Juegos Consola, en este caso MSF Ágil. La filosofía de este enfoque se encuentra resumida en el "Manifiesto ágil", en el cual se valora:

- Al individuo y las interacciones del equipo de desarrollo más que al proceso y las herramientas.
- Desarrollar *software* funciona más que conseguir una buena documentación.
- La colaboración con el cliente más que la negociación de un contrato.
- Responder a los cambios más que seguir estrictamente un plan.

MSF Ágil brinda mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del *software* con iteraciones muy cortas, según (Bauta, et al., 2007) es efectiva su aplicación en el proyecto Juegos Consola, en el cual los requisitos son muy cambiantes, el equipo es pequeño y se exige reducir

drásticamente los tiempos de desarrollo manteniendo la alta calidad. Además se cuenta con personal preparado, la cultura de la organización es flexible por lo que asimilan muy bien los cambios y se basan en nuevas técnicas. Para MSF Ágil el grado de adaptabilidad es muy alto y los cambios son bienvenidos.

Los principios para MSF Ágil que define (GUIDANCE, 2005) son:

- Aprender de todas las experiencias
- Alentar comunicaciones abiertas
- Trabajar hacia una visión compartida
- Invertir en calidad
- Permanecer ágil, esperar el cambio
- Otorgar poder a los miembros del equipo
- Concentrarse en agregar valor al negocio

1.2.1 MSF Ágil

MSF Ágil es la nueva propuesta de *Microsoft* en el mundo de procesos y prácticas ágiles de desarrollo de *software*. "*Microsoft Agile*" renueva al ya exitoso MSF V3.0, ahora la MSF V4.0 que aparece en agosto del 2006, implementa las mejores prácticas del mundo de desarrollo ágil de *software*, presenta dos principios adicionales: una mayor vinculación estrecha con los clientes y que todos los productos sean entregables. Utiliza *frameworks* descriptivos similares en muchos aspectos a MSF 3.0, pero la gran diferencia es que incluye dos procesos:

MSF para el desarrollo de Aplicaciones Ágiles (*Microsoft Solutions Framework* f*or Agile Software Development*) y MSF para el proceso de mejora CMMI (*Microsoft Solutions Framework* for CMMI *Process Improvement*). Al igual que la versión anterior define un equipo de trabajo, pero la ventaja es que aumenta la agilidad.

MSF Ágil tiene como principales características el ser altamente insistente, de planificación adaptable a los cambios y enfocado a las personas. MSF Ágil ofrece una guía sobre cómo organizar personas y proyectos para planificar, desarrollar y desplegar soluciones tecnológicas de *software* de una forma orientada al cambio. Es un proceso ágil y disciplinado de desarrollo de *software* y constituye un marco de

trabajo extensible y adaptable. Los principios base de MSF Ágil son básicamente los mismos principios de las metodologías Ágiles.

Las actividades atómicas o grandes actividades de este proceso se asignan y tracean mediante el concepto de *work ítems* (elementos de trabajo). Entre los tipos de *work ítems* predefinidos se encuentran: escenario, requerimiento de calidad de servicio, riesgo, tarea o *bug*. En otros procesos más elaborados se pueden encontrar éstos y/u otros *work ítems*, e incluso crear nuevos.

Las actividades en MSF Ágil están compuestas de 14 flujos de trabajo básicos, un flujo de trabajo es una actividad principal que se compone de otras actividades, 70 actividades específicas componen los 14 flujos, la mayoría de los flujos de trabajo se realizan por un mismo rol. Estos flujos son: definir la visión del producto, crear escenarios, crear requerimientos de calidad de servicio, planificar iteraciones, crear arquitectura de la solución, implementar tareas de desarrollo, construir un producto, probar un escenario, probar requerimientos de calidad de servicio, corregir *bugs*, cerrar *bugs*, *release* del producto, guiar el proyecto.

A diferencia de algunas metodologías tradicionales MSF Ágil es un proceso iterativo que incrementalmente va aproximando entregables de mayor fidelidad, que ha sido pensado para equipos reducidos. Es un proceso que no está dirigido únicamente por escenarios, sino también por requerimientos de calidad de servicio (como *performance* o seguridad). Y en lugar de centrarse en la arquitectura como lo hace RUP, asigna mayor relevancia al análisis de riesgos.

1.2.2 MSF Ágil para la gestión de proyectos informáticos.

En la actualidad son muchas las personas y organizaciones que no conocen acerca de los beneficios de las prácticas ágiles de desarrollo de *software*. Aunque han aumentado considerablemente los casos de éxito de proyectos que implementan estos enfoques.

El mejor modelo de desarrollo de *software* debe tener medios de *feedback* continuo y rápido, y "re-trazar" con la misma rapidez. Tiene que incluir métricas para los factores críticos del desarrollo y gestionar simultáneamente los relativos a la calidad y los relativos al coste. Debe abordar el desarrollo en pequeños incrementos. Aplicar controles de calidad desde las primeras fases. Trabajar con personas motivadas.

Incorporar un sistema de retro-información y mejora continua y estar orientado a los resultados finales, sin despistarse en las formas.

MSF Ágil es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Más importante que la tecnología que se usa en el proyecto informático, es el proceso que utiliza para controlar el mismo. MSF Ágil constituye el método más adaptable a proyectos de desarrollo de videojuegos. Proporciona guías de procesos y personal para ayudar a equipos y organizaciones a ser más exitosos en entregar soluciones de *software* a sus clientes.

Entregar soluciones de *software* de alta calidad, a tiempo y dentro del presupuesto, es un desafío para cualquier Proyecto. Si no se utiliza una metodología, una situación crítica o de desastre constituye una realidad, pero al mismo tiempo una metodología muy "protocolaria" (que exige una serie de pasos y documentos largos y detallados) como suelen serlo las denominadas "tradicionales", puede resultar inapropiado su uso, improductivo el proceso, provocar desgaste, costos excesivos y retrasos. Esto conduce a que una metodología detallista de proyectos puede ser tan peligrosa como no usar ninguna. MSF Ágil se sitúa en el centro de ambos extremos y se centra en aspectos importantes a manipular en un proyecto informático:

- La constitución y responsabilidades del equipo.
- Las fases y entregables del proceso de desarrollo.
- La vigilancia y mitigación continua de los riesgos.
- La administración distribuida de responsabilidades y la escalabilidad del modelo.

MSF Ágil establece la realización de reuniones, la creación de documentos y otros entregables, pero las exigencias son razonables y livianas, lo que favorece su utilización por parte del equipo. Además apoya la implantación de soluciones tecnológicas según (Colombia, 2000):

- Manteniendo siempre el enfoque al usuario. De esta manera asegura que la solución implantada realmente es lo que el usuario necesita.
- Proporcionando elementos valiosos a las inevitables preguntas: ¿Cuáles elementos van en el cliente?, ¿Cuáles en el servidor?

- Permitiendo un desarrollo basado en una filosofía de reutilización de múltiples componentes, lo que reduce el tiempo de desarrollo de nuevas aplicaciones, garantiza la interoperabilidad entre las mismas, y una mayor flexibilidad para incorporar los futuros cambios que se requieran.
- Estableciendo un equipo de trabajo balanceado, con tareas y objetivos claramente definidos, que permitan conocer el grado actual de avance del proyecto, y los riesgos que implica introducir modificaciones al mismo una vez iniciado su desarrollo.

MSF Ágil consta de roles, se denomina rol, según (MicrosoftOfficeOnline, 2007), al papel que ejerce un actor en una actividad o proyecto. Como plantea la (GUIDANCE, 2005), en MSF Ágil cada rol representa las necesidades específicas de su trabajo, representando los componentes del sistema relacionados con la producción, uso y mantenimiento del producto.

El conjunto de procesos al que denominamos MSF Ágil consta de roles, los cuales no necesariamente deben ser mapeados a personas diferentes, sino como lo estime el proyecto, es válido que una persona pueda desempeñar más de un rol.

Fig.1.3 Roles de MSF (GUIDANCE, 2005)

En el proceso MSF Ágil intervienen seis roles fundamentalmente: analista de negocio, jefe de proyecto, arquitecto, desarrollador, probador y administrador de *releases*.

En el epígrafe anterior se mencionaban de manera general las características fundamentales de MSF Ágil, entre ellas se manejó el concepto de *work ítems.*

Los *work ítems*, se pueden describir como registros de la base de datos que TFS utiliza para rastrear trabajos asignados y el estado del mismo. La base de datos común de *work ítems* y el almacén de métrica hacen posible dar respuesta a preguntas sobre la "salud" del proyecto en tiempo real.

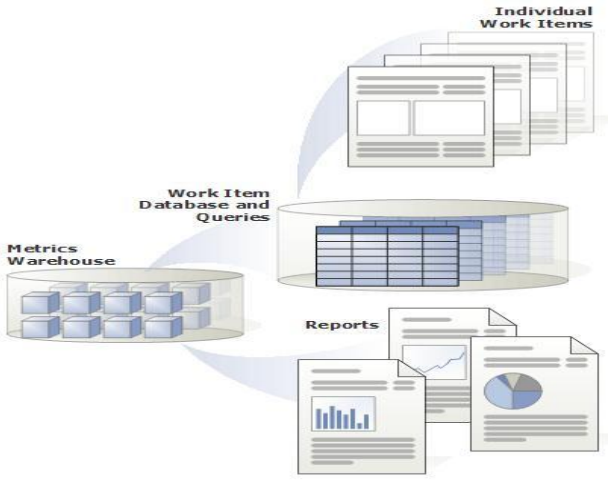

Fig. 1.4 *Work Ítems* y almacén de métricas (GUIDANCE, 2005)

El proceso MSF Ágil dispone de los *work ítems* siguientes:

- Escenario

En este *work ítems* se registra la descripción de la necesidad o solicitud del usuario, como una sola trayectoria de la interacción del usuario con el sistema. Un escenario registra las medidas específicas que se trazarán para que el usuario logre alcanzar una meta determinada.

- Requerimiento de Calidad de Servicio

Constituye un material resultante que documenta las características del producto final. Donde se mostrarán requisitos tales como: funcionamiento, carga, disponibilidad, tensión, accesibilidad, utilidad y capacidad de mantenimiento.

- Riesgo

Un riesgo constituye un evento o condición probable que puede dar resultados potencialmente negativos en el proyecto en el futuro. Por lo que se hace imprescindible para la administración de proyecto identificar y controlar los riesgos del mismo. El *work ítems* de tipo riesgo sigue las debilidades técnicas o de organización de un proyecto, así como la documentación correspondiente.

- Tarea

Una tarea es una acción independiente que debe realizar una persona o un grupo de personas. Este tipo de *work ítems* se utiliza para comunicar la necesidad de realizar un trabajo determinado. En dependencia del rol que se desempeñe, existen requisitos propios de una tarea. Además, se puede usar genéricamente una tarea para asignar el trabajo por separado dentro del proyecto.

- *Bug*

Un *bug* es un defecto o desviación entre el comportamiento esperado y el comportamiento observado en el producto. Este tipo de *work ítems* se emplea para comunicar un problema potencial que ha existido o que existe en el sistema, de manera tal que se vea claramente el impacto que puede traer dicho problema.

1.3 Modelos de MSF Ágil.

Anteriormente fueron enunciados los diferentes modelos de MSF (que pueden ser usados individualmente o combinados). Estos modelos son adoptados por la metodología ágil y es por ello que son aplicables a MSF Ágil, entre los principales modelos que contribuyen al éxito del proyecto encontramos: El Modelo de Equipos y el Modelo de Procesos.

1.3.1 Modelo de equipos.

Uno de los modelos centrales de MSF Ágil que contribuye al éxito del proyecto y que ha sido desarrollado para gestionar los equipos de trabajo, es el Modelo de Equipo, su diseño se dirige a mejorar el rendimiento del equipo de desarrollo, pues proporciona una estructura flexible para organizar los equipos

de un proyecto y puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles, por lo que podemos asegurar que no es un modelo rígido.

El MSF *team model* (modelo de equipos de MSF) fue desarrollado para compensar algunas de las desventajas impuestas por las estructuras jerárquicas de los equipos en los proyectos tradicionales, éste alienta la agilidad para hacer frente a nuevos cambios involucrando a todo el equipo en las decisiones fundamentales, asegurándose así que se exploran y revisan los elementos de juicio desde todas las perspectivas críticas, está orientado a la formación de equipos poco jerárquicos, pequeños y multidisciplinarios, con alta coordinación y comunicación para que en conjunto consiga los objetivos del proyecto.

En este modelo los miembros comparten responsabilidades y balancean las destrezas del equipo para mantenerse enfocados en el proyecto que están desarrollando. Comparten una visión común del proyecto y se enfocan en implementar la solución, con altos estándares de calidad y deseos de aprender.

Los seis roles que componen a este modelo, corresponden a las metas principales de un proyecto y son responsables por las mismas. Cada rol puede estar compuesto por una o más personas, la estructura circular del modelo, con óvalos del mismo tamaño para todos los roles, muestra que no es un modelo jerárquico y que todos los roles son igualmente importantes en su aporte al proyecto. Aunque los roles pueden tener diferentes niveles de actividad durante las diversas etapas del proyecto, ninguno puede ser omitido.

La comunicación se pone en el centro del círculo para mostrar que está integrada en la estructura y fluye en todas direcciones, como se muestra en la figura siguiente. El modelo apoya la comunicación efectiva y es esencial para el funcionamiento del mismo.

Fig.1.6 Modelo de Equipo (Fraile, et al., 2005)

Hay que tener en cuenta que al existir diferentes roles en un proyecto puede traer consecuencias y riesgos. Por ello es importante asignar apropiadamente estos roles a los miembros del equipo pues no todas las combinaciones para equipos reducidos son recomendables.

	Jefe de Proyecto	Arquitecto	Desarrollador	Tester	Analista de Negocio	Admin de Releases
Jefe de Proyecto		N	N	P	P	I
Arquitecto	N		N	I	I	P
Desarrollador	N	N		N	N	N
Tester	P	I	N		P	P
Analista de Negocio	P	I	N	P		I
Admin de Releases	I	P	N	P	I	

Fig. 1.7 Combinaciones para Equipos Reducidos (Fraile, et al., 2005)

P Posible **I** Inusual **N** No Recomendable

La meta principal del equipo de trabajo es entregar un sistema o solución de calidad, o sea, realizar un *software* y que el cliente quede satisfecho con éste, entregarlo en tiempo, gastando así la menor cantidad de recursos y coste, preparar al usuario para que sepa cómo usar el *software*, lo que implica la necesidad

de un mejor entendimiento entre los miembros del equipo y trabajar en una misma línea al tiempo que se responsabilizan con la(s) tarea(s) asignada(s). Por todas estas ventajas es que se aplica este nuevo modelo de equipos de MSF Ágil. Las metas de calidad sobre las cuales se concentran los esfuerzos del equipo de desarrollo son:

- Cumplir con las expectativas del usuario.
- Entregar el sistema o solución dentro de las restricciones del proyecto (tiempo, recursos, costos).
- Identificar todos los problemas o riesgos de importancia para el usuario y manejarlos de forma oportuna.
- Asegurar que el usuario final sepa cómo usar el sistema.
- Asegurar una implantación/replicación del sistema sin contratiempos.

El modelo de equipos se basa en la premisa de que cada función presenta una perspectiva única en el proyecto y que ninguna persona por sí misma puede representar de una manera satisfactoria todos los objetivos de calidad de todas las funciones.

Las funciones del equipo de MSF Ágil comparten responsabilidades en muchos aspectos de la administración del proyecto como por ejemplo, la administración del riesgo, del tiempo, de la calidad, del planeamiento, de la programación, de la selección de los miembros del equipo y de los recursos humanos.

1.3.2 Modelo de Procesos

Los modelos de procesos establecen el orden de las actividades del proyecto, representando completamente el ciclo de vida de éste, su aplicación es imprescindible para el desarrollo de un proyecto de *software*, ya que se enfatiza en todo el proceso sin centrase exclusivamente en el producto. Además incluye algunas actividades que no se relacionan exclusivamente con el proyecto.

MSF Ágil también tiene su modelo de procesos que hace posible avizorar y controlar el cambio. MSF Ágil se basa en metas, las metas son puntos a terminar en el proyecto y éstos pueden ser revisados. El modelo de procesos MSF combina los mejores principios del modelo en cascada y del modelo en espiral, la claridad que plantea el primero de éstos y las ventajas de los puntos de transición del modelo en espiral. Además, combina una secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones de tecnologías de la información (IT por las siglas en inglés de *Information Technology*).

Fig. 1.8 Procesos de MSF Ágil por fases (Fraile, et al., 2005).

El modelo de procesos de MSF Ágil consta de cinco fases, éste suministra una imagen más clara del estado de dichos productos en cada etapa sucesiva, a través de su estrategia iterativa en la construcción de productos del proyecto. El equipo puede identificar con mayor facilidad el impacto de cualquier cambio y lidiar con él de manera efectiva, minimizando los efectos colateralmente negativos y optimizando los beneficios.

Los principios del modelo de procesos de MSF Ágil son:

- Trabajar con una visión en común.
- Mantenerse ágiles, esperando cosas que puedan cambiar.
- Concentrarse en la entrega de valores de negocios.
- Fomentar la comunicación abierta.

Este modelo es iterativo, se basa en obtener una visión de los próximos proyectos o sistemas, prioriza el análisis de riesgos e implementa incrementalmente con puntos de revisiones frecuentes, que relaciona el trabajo del equipo con las expectativas de los usuarios a lo largo de todo el proyecto. Los cuatro puntos de revisión principales están precedidos por una fase principal dentro de la cual ocurren puntos de revisión internos y se pueden generar productos entregables. (Jacobson, et al., 1999)

Según (Jacobson, et al., 1999) los puntos de revisión no necesariamente son puntos de congelación. Cada punto de revisión establece un punto de partida que permite en forma confiable que el equipo planee y continúe hacia el siguiente punto de revisión.

El modelo de procesos de MSF Ágil se aplica para encontrar un balance en cuanto a alcance, tiempo y recursos de un proyecto, a su vez puede ser aplicado a una amplia variedad de proyectos sin pérdida del flujo del proceso.

1.4 MSF Ágil y el entorno integrado de desarrollo *Visual Studio* (IDE).

Uno de los aspectos más interesantes acerca del proceso MSF Ágil, es el hecho de que ya viene integrada a la plataforma de desarrollo de *Visual Studio* 2005 *Team System*, el cual permite crear proyectos para el trabajo en equipos (*Team Projects*).

1.4.1 *Visual Studio Team System (VSTS)*

El VSTS es una plataforma para un conjunto de herramientas extensibles, integradas y productivas 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. Estas herramientas se integran con el ambiente de desarrollo integrado del *Visual Studio* (IDE), *Office*, los servicios de *SharePoint*, y los servicios de reportes del SQL *Server* 2005. Estos servicios ofrecen una personalización y extensibilidad casi ilimitada.

VSTS viene configurado con dos procesos MSF: MSF para el desarrollo ágil y MSF *for* CMMI para el proceso de mejora, y por medio de éste es posible contar con una serie de plantillas y guías adaptadas a cada proceso y orientadas a los roles definidos en estos. No obstante el VSTS permite modificar estas plantillas para crear versiones personalizadas.

Una plantilla de proceso es un mecanismo de VSTS que define tanto la configuración como el contenido inicial del proyecto de equipo, a través de un conjunto de *work ítems* por defecto, consultas de *work ítems*, plantillas de artefactos, reportes, grupos de seguridad y documentos de la orientación del proceso. Cada plantilla de proceso está basada en un proceso de desarrollo de *software* que va acompañado de un enfoque muy particular acerca de cómo desarrollar y mantener *software*.

Entre sus principales características encontramos, la integración al *Visual Studio* 2005 debido a la gran resistencia al cambio del equipo de desarrollo, ya que las nuevas políticas nunca son agradables. Dada esta integración se reduce considerablemente la curva de aprendizaje en la utilización de las herramientas proporcionadas, contribuyendo a la reducción del tiempo en que se llevan a cabo estas actividades. La extensibilidad constituye su característica principal, pues las diferencias existentes entre cada persona, equipo de trabajo y empresa, exigen la capacidad de adaptar esta solución a las necesidades particulares de cada empresa, equipo y persona, y de esta forma lograr una completa integración y rápida implementación.

VSTS implementa seguridad a varios niveles, con un conjunto de permisos por proyecto. Se enfoca particularmente en el requerimiento de información veraz y oportuna que necesita el Jefe del Proyecto. Utilizando un sistema de reporteo, esta información es presentada de manera entendible y en tiempo para una mejor toma de decisiones. Los reportes de proyectos que se integran en VSTS son visibles a través de *Windows SharePoint Services* (WSS); *Microsoft Excel* y *Microsoft Project.*

1.4.2 *Visual Studio Team Fundation Server* (VSTFS)

Como se describe en (Bauta, et al., 2007) el TFS es una colección de las tecnologías de colaboración que apoyan un esfuerzo del equipo para entregar un producto determinado, aunque pueden también ser utilizadas en otros tipos de proyectos.

Las vistas sobre el trabajo de desarrollo se generan en función de los roles de los miembros involucrados en el proceso de desarrollo (arquitecto, desarrollador y probador). El TFS, ya sea a nivel de pantallas de configuración o manejando las APIs del sistema, permite una total personalización de su entorno, lo cual constituye una ventaja de esta herramienta.

El manejo de la seguridad en TFS puede considerarse otra de sus ventajas, está basada en usuarios y grupos que se pueden manipular para lograr implementar un modelo de seguridad que permita a los usuarios acceder a los datos y la funcionalidad que ellos requieran, mientras se protege la información principal del proyecto.

El modelo de seguridad de TFS es un poco complejo ya que aparte de los permisos asignados dentro del VSTS, está la asignación de privilegios y usuarios en otros aplicativos. Existen tres grandes grupos para otorgar permisos y estos son:

1. Grupo de TFS
2. Grupo de Servicios de WSS (*Windows SharePoint Services*)
3. Grupo de Servicios de Reportes

Esta tarea es propensa a errores, para evitarlos se puede utilizar el aplicativo llamado Herramienta para la Administración de TFS (*Team Foundation Server Administration Tool*) para toda la comunidad de desarrollo.

VSTFS proporciona una plataforma cohesiva para el desarrollo de *software* y para la colaboración en equipo, esta herramienta que se considera el lado del "servidor" de VSTS ,habilita el marco de trabajo para éste último, incorpora elementos para la gestión de procesos, seguimiento de los *work ítems*, un portal de proyectos (basado en WSS), elaboración de reportes, pruebas y controles de equipo con la generación de reportes de modo centralizado, un almacén de datos de métricas, el control de código fuente, la automatización de versiones, recomendaciones sobre procesos MSF y servicios de integración.

Un aspecto muy importante es que brinda un soporte computacional para el control y seguimiento de procesos de desarrollo. Entre estos procesos se incluyen MSF para el desarrollo ágil del *software* (para estrategias más ligeras e iterativas) y MSF para el proceso de mejoras de CMMI (aproximaciones más estructuradas y rigurosas). Estos procesos son personalizables a través de campos, formularios, estados y reglas utilizando el asistente de Dirección de Plantillas de Procesos y la plantilla CMMI mencionada anteriormente, es un súper conjunto de la plantilla *Agile*. El asistente de creación de proyectos ayuda a lanzar proyectos y el "editor de procesos" permite a los usuarios enmarcar los proyectos en el contexto de las mejores prácticas.

Constituye 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, pero no serviría de mucho contener este tipo de información si no se tiene la capacidad de explotarla correctamente.

El VSTFS utiliza **SQL *Server 2005*** para almacenar todo lo relativo a los *work ítems*, atributos de calidad, pruebas, resultados de pruebas y resultados del *build*. TFS utiliza después SQL *Server Analysis Services* para agregar y analizar los datos y dirigir los reportes. Los reportes que crea la plantilla del proceso o los miembros individuales del equipo con el Diseñador de informes de *Microsoft Excel* o *Visual Studio* 2005 están disponibles en SQL *Server* 2005 *Reporting Services* y en el sitio de reportes del equipo.

TFS utiliza los componentes de *Microsoft*, tales como SQL *Server* 2005, *SharePoint* y *Reporting Services*, con el objetivo de brindar las siguientes funcionalidades:

Control de Código Fuente

El control de versiones de TFS es una herramienta de gestión del cambio que permite administrar activos *work ítems*. Incluye funciones tales como *changesets (*conjunto de cambios*)*, ramas, fusiones y comparación de archivos, que son propias de todo sistema de control de código fuente de escala empresarial; así mismo permite efectuar protecciones atómicas y desarrollar remotamente.

Las fuentes de la solución, 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, permitiendo así obtener una transparencia sobre el mismo, definiendo de manera rápida cuestiones como el último cambio, cuál fue el mismo y quién lo produjo.

Seguimiento de *Work Ítems*

Estos *work ítems* permiten la asignación fácil y rápida de tareas, el flujo dinámico de información entre los equipos que conforman el proyecto y muestran información útil que permite definir elementos como la velocidad del proyecto de manera confiable. Todos los *work ítems* se localizan en una única base de datos para permitir una coordinación más precisa.

Reportes

El jefe de proyecto debe ser capaz de responder diariamente las siguientes preguntas acerca del curso del negocio:

¿Cuánto trabajo se ha hecho y cuánto queda por hacer?

¿Cuán productivo es el equipo?

¿Cuál es la calidad del software que estamos produciendo?

¿Con qué efectividad se está encontrando, reparando y cerrando los defectos?

¿Cuántas tareas falsas y reparación de defectos tenemos?

¿Estamos encontrando y tratando correctamente los defectos?

¿Cuán rápido se puede avanzar sin afectar la calidad?

Cada pregunta es relevante en muchas escalas para los días dentro de una iteración, para las iteraciones dentro de un proyecto y para los proyectos dentro de un programa. Las preguntas son relevantes además para todos los tipos de escenarios y otros requerimientos, tareas, *bugs* y otros.

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 *Reporting Services* mostrando datos útiles a las personas que deberán tomar decisiones en el proyecto.

Los reportes constituyen una de las características más poderosas de VSTS ya que permiten al administrador del proyecto dar seguimiento a la información acerca del estado y de las tendencias del proyecto de equipo. TFS incluye un almacén de datos en el que se recopilan los datos operativos que provienen de los *work ítems*, control de versiones, los *build* del producto y resultados de las pruebas. Este almacén es empleado por los SQL *Reporting Services* para producir los reportes. Una de las enormes ventajas de usar VSTS, desde la perspectiva de administración de proyectos, es que no es necesario correlacionar manualmente datos provenientes de varias fuentes.

Team Portal

Provee 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 estado del proyecto, guías sobre la realización de actividades, entre otras.

El VSTFS es fundamental si se requiere mejorar la previsibilidad de entrega del proyecto de desarrollo, si es necesario integrar una mayor organización, o para reducir el tiempo dedicado a preparar informes y reuniones acerca del estado del proyecto, todo esto encaminado a obtener un producto con la calidad requerida.

Team Project

Team Project es un contenedor que mantiene información acerca de cada paso del ciclo de vida de desarrollo de *software* en un repositorio central dentro de TFS. Los proyectos de equipo se componen de una serie de *work ítems*, piezas de código, casos de prueba, productos de trabajo, métricas, entre otros, los cuales son empleados para dar seguimiento al trabajo de un proyecto.

Existen diferencias substanciales entre *Team Project* y un proyecto de *Visual Studio*, mientras que este último sirve para organizar código, el primero sirve para organizar todo el esfuerzo de desarrollo de *software*.

Team Explorer

Para crear un proyecto de equipo se emplea una ventana de la IDE de *Visual Studio* llamada *Team Explorer*. Esta ventana permite navegar y administrar todos los elementos de un proyecto de equipo tales como la estructura del equipo de trabajo, el portal del proyecto, el repositorio de control de código fuente, la base de datos de *work ítems* , documentos, reportes y plantillas.

1.4.3 Interactividad entre VSTS y VSTFS a través de los procesos MSF Ágil.

El VSTS proporciona una plantilla de la estrategia para fijar el proceso. Esta plantilla contiene los *work ítems*: escenario, requisitos de calidad del servicio, tareas del desarrollo, tareas de prueba, riesgos, ajustes de seguridad, dirección de proceso, reportes de proceso, políticas de los registros de la fuente, entre otros. A su vez, *Microsoft* proyecta las plantillas y los servicios del proyecto sobre un portal/*Windows* y *SharePoint* localiza la plantilla configurada para apoyar los procesos de MSF.

MSF Ágil proporciona la dirección integrada al plan, los requisitos de estructura, arquitectura, diseño, desarrollo, y prueba el proyecto. Esta dirección se inserta totalmente de modo que aparezca justamente diseñada como una extensión integrada al *Visual Studio*.

MSF versión 4.0 es el último *framework* de la serie, basado en las mejores prácticas de *Microsoft* y sus clientes. Presenta un metamodelo descriptivo y dos plantillas de procesos prescriptivos que implementan MSF 4.0 en VSTS. Juntas son herramientas muy poderosas para equipos de todo tipo. MSF proporciona la guía de proceso y VSTS un conjunto de herramientas integradas y extensibles.

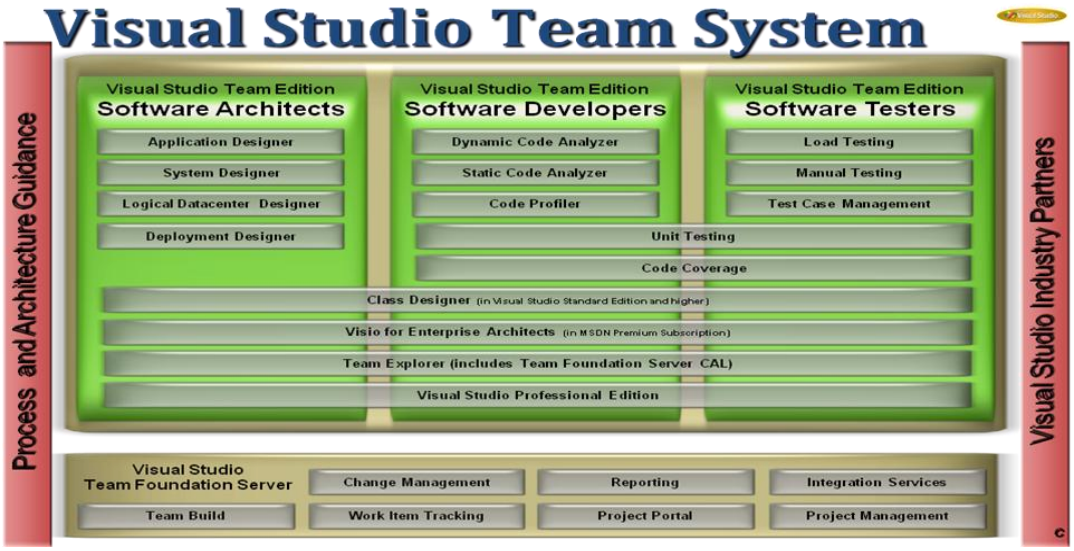

Fig. 1.9 *Visual Studio 2005 Team Suite* (Fraile, et al., 2005).

La versión *Suite* del *Visual Studio* brinda la posibilidad al equipo de desarrollo de establecer la comunicación y colaboración ideal. Es importante conocer el contenido de cada edición, que plantea (Guckenheimer, 2006) para tener un mejor dominio de estos productos:

Edición para desarrolladores de *software*: Proporciona las herramientas de desarrollo avanzado que permiten a los equipos construir servicios y usos confiables.

Edición para arquitectos de *software*: Proporciona a los diseñadores visuales permitir a arquitectos, jefes de explotación y liberadores diseñar las soluciones servicio-orientadas que se pueden validar contra los ambientes operacionales.

Edición para Probadores de *software*: Proporciona las herramientas de prueba de carga avanzada que permiten a los equipos verificar el funcionamiento antes del despliegue.

Edición *Suite*: Combina cada uno de los productos anteriores en una herramienta de gerencia asequible al ciclo de vida, que trata las necesidades de que existan roles de múltiples funciones en una organización.

Todos estos productos son soportados por VSTFS, un servidor extensible de colaboración que permite a todos los miembros del equipo manejar y seguir el progreso y la "salud" de los proyectos.

Con la buena puesta en marcha del VSTS en proyectos de desarrollo de videojuegos se puede reducir la complejidad de entrega de las soluciones, se facilita la colaboración entre todos los miembros de un equipo del *software*, dada la necesidad de cumplir con el tiempo de desarrollo establecido para cada proyecto y de asegurar la pre-visibilidad y la confiabilidad del proceso. Permite modificar y extender el VSTS para requisitos particulares, en conjunto con las herramientas y procesos del mismo.

Además posibilita que los socios y clientes extiendan el VSTS con sus propias herramientas internas o comerciales y estructuras de proceso, mientras que la plataforma en la cual se construye el VSTS y los servicios de base del TFS permite a terceros extender el sistema y compartir reportes, eventos, grupos y permisos, administración y capacidades para la navegación. Cada una de las herramientas del VSTS puede ser extendida con APIs o servicios propios del equipo de desarrollo involucrado.

1.4.4 *Microsoft Office SharePoint Server* (MOSS) 2007

MOSS representa la siguiente generación de *SharePoint Portal Server* (SPS) creada a partir de *Windows SharePoint Services* (WSS). El gráfico circular que se muestra en la figura 1.10, representa de forma sencilla el trabajo en conjunto de MOSS y WSS. Las áreas en verde pertenecen a WSS y debido a que MOSS se basa en WSS, todo el gráfico refleja el trabajo de MOSS. MOSS proporciona concretamente el

portal, la búsqueda, la administración del contenido empresarial, el proceso de negocio y formularios, y ejemplos de inteligencia empresarial.

Fig. 1.10 Servicio creado a partir de MOSS y WSS (Rizzo, 2007).

Orientada a desarrolladores con experiencia en ASP.NET 2.0 y conocimientos sobre *Windows SharePoint Services* 3.0, esta exhaustiva guía ha sido diseñada para que sirva de recurso útil y conciso, proporcionando respuestas rápidas y eficaces en su día a día.

Microsoft Office SharePoint Server 2007 admite todas las intranet, extranet y aplicaciones *Web* de una empresa en una sola plataforma integrada en lugar de depender de sistemas fragmentados independientes. Además, este servidor de colaboración y administración de contenido proporciona tanto a profesionales de IT (Tecnología de la Información) como a programadores la plataforma y las herramientas necesarias para la administración de servidores, la extensibilidad de aplicaciones y la interoperabilidad. También incluye nuevas características de colaboración actualmente inasequibles a VSTS, incluyendo las *wikis*, los *blogs* y otros. Aunque VSTS no puede tomar la ventaja directa de estas nuevas características, la integración con *SharePoint* está en el VSTS para una versión futura, que ampliará sus capacidades de colaboración.

Principales ventajas que *Microsoft Office SharePoint Server* 2007 puede aportar a la organización:

- Aumentar la productividad de los empleados al simplificar las actividades cotidianas.
- Ayudar a cumplir los requisitos legales mediante el control exhaustivo del contenido.
- Administrar y replantear el contenido de forma eficaz para obtener mayor valor empresarial.

- Conectar a las personas con información y recursos especializados, protegiendo la información confidencial.
- Agilizar los procesos de la organización y mantener el control del entorno de formularios electrónicos.
- Facilitar la toma de decisiones mejor fundamentadas presentando la información empresarial importante en una sola ubicación central.

MOSS ofrece características mejoradas para buscar repositorios empresariales, a partir de la plataforma de búsqueda proporcionada originalmente en *SharePoint Services* 2003. MOSS 2007 sigue rastreando repositorios de recursos compartidos de archivos, sitios *web*, *Exchange Server* y *Lotus Notes*, que se consideran repositorios de datos sin estructurar.

MOSS puede buscar repositorios estructurados a través de una nueva tecnología denominada Catálogo de datos profesionales (BDC), así como buscar usuarios y expertos mediante la nueva red de conocimiento.De forma predeterminada, *SharePoint* crea una página de perfil para cada entidad de negocio del sistema del servidor. Estas páginas de perfil se pueden personalizar de la misma forma que otras páginas en *SharePoint*. Ofrece una sólida solución de administración de contenido empresarial al aumentar la compatibilidad de administración de registros, contenido *web* y formularios electrónicos.

1.5 Herramientas de uso libre útiles para TFS

En junio del 2005 se dio a conocer un *plug-in*, *Visual MainWin para* J2EE *Developer Edition*, disponible de forma gratuita, está diseñado para unir los desarrollos de *Visual Studio* a Linux y J2EE. Referido como *Grasshopper,* ha sido diseñado por *Mainsoft*, que afirma que es el primer IDE basado en *Visual Studio* para Linux.

Este añadido permite a los desarrolladores, depurar y desplegar aplicaciones *web* y servicios *web* para *Windows*, Linux y plataformas habilitadas con Java utilizando el *software Visual Studio*, C# y *Visual Basic*.*Net*. La oferta está dirigida a los desarrolladores individuales y pequeños grupos. Con *Grasshopper*, los desarrolladores de *Visual Studio*.*Net* pueden realizar mejoras y librerías de clases, entre otras cosas.

Actualmente, aparte de todo lo que se puede hacer con TFS, tanto desde el GUI, como desde herramientas de línea de comando, hay algunas herramientas que se están desarrollando, en forma de uso libre, y que ciertamente ayudan bastante a la hora de trabajar con TFS.

Según (*Team Fundation Server Administration Tools*, 2007) **TFS *Admin Tool*:** uno de los aspectos no muy cómodos de hacer en TFS, es la administración de permisos de los usuarios, ya que son tres los sitios donde tenemos que dar permisos, en TFS, en *SharePoint*, y en *Reporting*, con esta herramienta, y desde una interfaz gráfica única para los tres, podemos gestionar los permisos para los tres sistemas.

TFS *Admin Tool* permite al administrador de TFS:

- Agregar, eliminar, y modificar múltiples permisos de usuarios desde el *Team Project* y aplicarlos al mismo tiempo.
- Identificar los permisos que faltan desde el *SharePoint* o SQL *Reporting Services* y corregirlos.
- Ver los registros de cambio de permisos que han ocurrido.
- Definir qué permisos del *SharePoint* y SQL *Reporting Services* deben ser usados automáticamente cuando se crea un nuevo usuario del TFS.

En septiembre del 2007 fue liberado un conjunto de herramientas de importancia para el TFS, el ***Visual Studio* 2005 *Team Foundation Server Power Tools*** es un conjunto de accesorios, herramientas y utilitarios para línea de comando que mejoran la experiencia del usuario del TFS. Este lanzamiento incluye características de las versiones anteriores del *Power Tools* y agrega dos magníficas herramientas: el *Team Foundation Server Analyzer* y el *Work Ítem Templates*.

***Power Tool* contiene:**

- *Team Foundation Power Tool Commands*: Es una herramienta de línea de comandos que en muchos de los casos se presenta en una interfaz gráfica para el usuario.
- *Process Template Editor:* Sirve para crear y modificar plantillas de procesos para TFS.
- *Custom Check-In Policy Pack:* Provee al TFS una lista de nuevas políticas para los *check-in* dentro de *Team System*.
- *Test Tool Build Task*: Esta herramienta evita la especificación de los archivos metadatos para ejecutar un *Build* en TFS, al correr la lista de pruebas que se tiene dentro del proyecto.

- *Team Foundation Server Best Practice*: Es una herramienta de diagnóstico.
- *Work Ítem Templates*: Este paquete instala una serie de plantillas de *work ítems* en el servidor, las cuales soportan características de edición, captura, de aplicación y se convierten en las plantillas por defecto de la herramienta.

Una de las herramientas de este grupo de mayor importancia para el TFS en el proyecto es ***Process Template Editor*.** Según (*Imaginet Resource Corp*) cuando se hacen las modificaciones en las plantillas de creación de proyectos hay dos opciones, o modificarlas manualmente y validarlas contra los esquemas que proporciona *Microsoft* en el *Visual Studio 2005* SDK (*Software Development Kit*) o se descarga el *Process Template Explorer* para hacer ahí todos los cambios que se deseen, permite modificar las plantillas y los tipos de *work ítems*, las carpetas de *SharePoint*, informes de *Reporting*, también listas globales de TFS, en resumen, todo lo necesario para personalizar el entorno.

La instalación incluye documentación aparte para el VSTS *Process Editor*, el cual contiene una guía de usuario y un archivo *readme* que incluye las cuestiones conocidas. Para utilizar el *Team System Process Editor* es necesario tener instalado el *Domain-Specific Lenguage Tools* para el *Visual Studio* 2005 *Redistributable Components*.

Esta herramienta, de desarrollo compartido, permite definir nuevos *work ítems*, editar los campos asociados a cada uno, manejar los estados y las transiciones admitidas. Los formularios pueden diseñarse gráficamente, asignando y modificando la distribución de los campos. De manera oculta el *Process Template Editor* mantiene los documentos XML correspondientes a cada elemento. Posee funciones para importar y exportar las definiciones entre un proyecto de TFS y un documento XML. Además pueden editarse, abriéndolas de un proyecto directamente (del *server*). En este último caso, en algunas ocasiones la operación falla, terminando con una excepción. Cuando esto sucede ya no es posible abrir ese tipo de *work ítem* desde ese proyecto de TFS, por ello es recomendable exportar la definición a XML. De esta manera, si la apertura desde el proyecto del *server* no es posible, siempre puede trabajarse sobre el XML de respaldo y luego importarlo al proyecto.

1.5.1 *NTeam*, la alternativa libre a VSTS

Se encontró en (Navegapolis.com, 2005) que se comenzó hace tres años la planificación de un sistema *open-source* como alternativa a MVSTS. Esto se llevó a cabo por *Alan Stevens*, *Jason Benthley* junto con

un equipo de 33 programadores y gestores de proyectos. El proyecto se llama *NTeam* e integrará herramientas y aplicaciones *open-source* ya existentes para los módulos de pruebas y desarrollo (*NUnit* y *NAnt*).

Al igual que el VSTS de *Microsoft*, *NTeam* también cubrirá las diversas fases del ciclo de vida, si bien en su primera versión es previsible que implemente algunas funcionalidades menos. *NTeam* permitirá disponer de forma gratuita de un marco de gestión y seguimiento de proyectos, adecuado para entornos ágiles.

CAPÍTULO II. ESTADO ACTUAL DEL PROCESO DE DESARROLLO DE VIDEOJUEGOS VS. ESTADO DESEADO.

Introducción

En el mundo de los procesos y prácticas de desarrollo de *software*, existen constantes enfrentamientos entre aquellos que defienden el enfoque ágil y los que defienden la idea de que el éxito de un proyecto de desarrollo de *software* se alcanza exclusivamente a través de metodologías formales o rígidas, con elevados niveles de definición. Esto influye en gran medida en las decisiones a la hora de seleccionar la metodología a utilizar, con dependencia directa de las características o cualidades del equipo de trabajo, la cultura de la organización, la frecuencia de los cambios del entorno y la aceptación del usuario final.

En el presente capítulo se analizan los aspectos críticos del proyecto Juegos Consola, se describe su situación actual y los resultados obtenidos. Además, se realiza una comparación exhaustiva, enfrentando esta situación existente donde no se utiliza ninguna metodología cabalmente, contra el proceso de desarrollo de videojuegos guiado por MSF Ágil, con el objetivo de encontrar las deficiencias existentes y las mejoras potenciales al adoptar las prácticas que más benefician al proyecto.

2.1 Estado actual del proyecto Juegos Consola.

En el curso 2004-2005 se crea el proyecto Juegos Virtuales perteneciente a la Facultad 5, con el objetivo de desarrollar videojuegos para colaborar con los parques de diversiones existentes en el país que carecen de este tipo de servicio, y de esta forma cumplir con el plan del país dirigido a la mejora de estas instalaciones. En el año 2007 el proyecto es nombrado Juegos Consola por las características específicas de la producción.

En el curso actual 2007-2008 en el proyecto no se pone en práctica a cabalidad ninguna metodología en el desarrollo de videojuegos, se obtienen artefactos de RUP y se aplican algunas prácticas de XP. Dadas las características propias del desarrollo de videojuegos existe una documentación excesiva.

El equipo se organiza de forma jerárquica, está constituido por 20 integrantes y varios de ellos juegan un mismo rol. No existe una buena coordinación del equipo, y el líder de proyecto no cuenta con herramientas de apoyo para el control del mismo.

El proceso de desarrollo de videojuegos se centra en la arquitectura, se añaden funcionalidades en cualquier momento del desarrollo, el equipo no está capacitado para afrontar los cambios que exige el proceso de desarrollo de videojuegos. En la realización del mismo, el proceso se dirige por casos de uso. La construcción se lleva a cabo de manera incremental, pero no iterativa, y actualmente la integración se realiza al final del proceso de desarrollo. Los módulos independientes son escritos en C++ haciendo uso del entorno integrado de desarrollo (IDE) *Microsoft Visual Studio* 2005 y para el control de versiones el equipo se apoya en *Microsoft Visual SourceSafe* 6.0.

No se realizan pruebas unitarias durante el proceso de desarrollo actual de videojuegos.No se tienen en cuenta actualmente métricas descriptivas que midan el estado actual del proyecto en cuanto a sus elementos críticos, en caso de empleo de métricas, es de manera prescriptiva.

Lo anteriormente expuesto ha implicado, retraso en la entrega del producto final. Esto conlleva a la búsqueda de buenas prácticas que ajustadas a las necesidades propias del proyecto permitan revertir la situación actual existente en el proyecto Juegos Consola perteneciente a la facultad 5.

2.1.1 Documentación excesiva vs. Conocimiento implícito

En la práctica en el proyecto Juegos Consola, se realiza una gran cantidad de documentación que se utiliza en los diferentes procesos, por documentación se entiende la realización de la especificación de cualquier artefacto, modelos gráficos, diferentes planes, documentos de ambiente de desarrollo, minutas de reuniones, presentaciones, otros no relacionados directamente con el código ejecutable, datos, plantillas de las pruebas del *software* que se está construyendo, roles y responsabilidades, además de archivos de ayuda, manuales, capacitación y otros materiales que son parte de valor del producto que será entregado.

Toda la documentación que se realiza debe tener un propósito claro y justificado, por lo general se enfoca a la finalidad del contrato, el consenso, la arquitectura, la aprobación, la fiscalización, la delegación o el seguimiento del trabajo. Hay que tener claro que a veces mucha documentación no es la forma más efectiva para alcanzar el objetivo. Según (Guckenheimer, 2006) esto es favorable o necesario cuando el equipo es muy grande y está dividido en partes separadas geográficamente, donde el dominio de los conocimientos varía, y existe un gran número de interesados que necesitan revisar los diseños, además en estos casos en los cuales por razones de distancia las discusiones no pueden ser cara a cara y se efectúan de manera electrónica. En estas circunstancias sí constituye una necesidad una documentación más explícita, detallada y suficiente para la comprensión y seguimiento del proceso de desarrollo y el control del trabajo del equipo. Pero evidentemente éste no es el caso del proyecto.

A la manera de MSF Ágil según (Guckenheimer, 2006) cuando se tiene un equipo pequeño, cohesionado y ubicado en un mismo local con un fuerte dominio del conocimiento, puede ser fácil de mantener y alcanzar el objetivo sin documentación inútil y de esta manera no incurrir en el gasto innecesario de esfuerzos y recursos. Un equipo con estas características es posible que necesite poco más que una enunciación de la visión, un listado de escenarios y una frecuente comunicación cara a cara con el cliente. La clave es escribir documentación exclusivamente para su audiencia, enfocado a los propósitos específicos del proyecto y luego dejar de escribir documentación. De esta forma acumular el esfuerzo y recursos para emplearlos en la codificación. En otras palabras emplear mayores esfuerzos y recursos en los productos de trabajo que le ofrecen un valor directo y sustancial al cliente. En caso de que con el avance del proyecto se descubra que la documentación es insuficiente para alcanzar el objetivo, ésta se actualiza. Pero sólo donde se necesita.

2.1.2 Estructura jerárquica vs. Modelo de equipo que propone MSF Ágil.

La organización de un equipo puede tomar muchas formas. Por ejemplo, puede trabajar con lazos estrechos o relativamente dispersos, puede organizarse en una pirámide jerárquica o con esquemas más o menos horizontales, además de asumir un conjunto estricto de reglas o regirse por orientaciones informales.

La estructura jerárquica que organiza al equipo de trabajo del proyecto Juegos Consola hace que el trabajo sea dirigido por el jefe de proyecto, éste supervisa cada detalle y toma todas las decisiones en el

proyecto. Las reglas que se establecen se diseñan para promover la estabilidad y continuidad de trabajo del equipo, éstas se mantienen bajo un control estricto de forma que no se desvíen de lo establecido. Esta estructura es de tipo piramidal donde los roles están bien específicos para cada nivel, de forma tal que no todos los roles en el proyecto Juegos Consola tienen la misma importancia dentro del equipo, es por esto que las actitudes críticas e independientes son tomadas como signo de deslealtad y no son toleradas. La forma en la que se estructura el equipo de trabajo es predictiva favoreciendo la planificación, por tanto se resiste al cambio y a las innovaciones. Según se registra en el expediente del proyecto se han definido muchos roles.

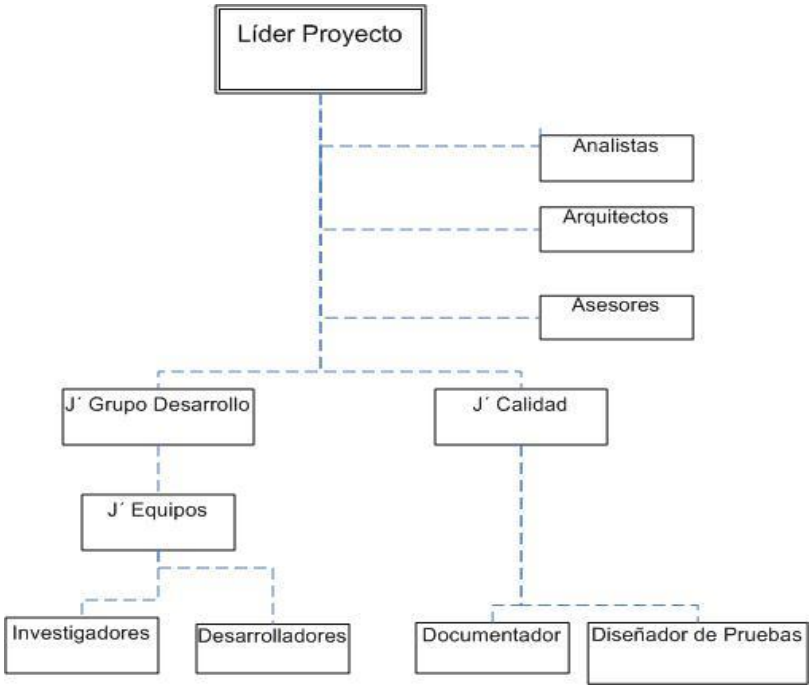

Fig. 2.1 Estructura jerárquica del Proyecto Juegos Consola

El proyecto Juegos Consola debe caracterizarse por ser innovador, por tanto, se necesitan orientaciones más que reglas, esta rigidez y control en la organización del equipo provoca la resistencia al cambio. La estructura actual del equipo no se adapta a las necesidades y características del proceso de desarrollo de videojuegos del proyecto Juegos Consola.

Por otra parte, si se adopta en el proyecto el modelo de equipo que propone MSF Ágil y las ventajas de su integración como proceso al VSTS, se solucionan algunas de las deficiencias de la estructura piramidal pues proporciona una estructura flexible para organizar el equipo, escalándolo según las personas disponibles para conformar el equipo, y de esta manera se aleja de un modelo rígido. Ésta se considera una "estructura abierta", que se orienta a promover la innovación y el cambio a través de la creatividad necesaria en el proyecto, posibilita la coordinación de iniciativas individuales, la colaboración abierta a través de la discusión y la negociación.

El modelo de equipo que propone MSF Ágil comparte flexiblemente los roles y las responsabilidades, de forma que todos los roles tienen la misma importancia dentro del equipo, de allí que se denomine "equipo de iguales" o "equipo de pares". La información fluye fácilmente entre los miembros del equipo, se promueve la combinación de diferentes puntos de vista y alienta la agilidad para hacer frente a nuevos cambios involucrando a todo el equipo en las decisiones fundamentales, por esto los individuos se ven incentivados a participar activamente en los proyectos al sentirlos como propios. Los rasgos adaptativos del modelo posibilitan el aprovechamiento completo de la información y la integración de las contribuciones de todos los miembros del equipo.

El modo de operación del modelo se centra en sesiones de resolución de problemas cara-a-cara en las cuales se establecen ciertos roles claves. Estos roles no necesariamente son fijos, sino que pueden rotar entre los miembros del equipo a medida que avanza el proyecto o de acuerdo al problema planteado. Son pocos roles, más genéricos y cada miembro del equipo puede desempeñar múltiples roles, permitiendo que todo el equipo tenga una visión común del proyecto.

De esta manera el equipo se enfoca en implementar la solución, con altos estándares de calidad sacando provecho de las ventajas del VSTS. Esta herramienta define roles por funciones específicas dentro del proyecto, permitiendo la modificación de los roles predefinidos según características específicas del equipo; adapta la experiencia del usuario y las diversas capacidades que posee a cada rol en particular. Para que los miembros del proyecto cambien con frecuencia funciones de trabajo de manera ágil el VSTS proporciona menús constantes e interfaces de usuario que facilitan las transiciones entre los roles.

2.1.3 Características del ciclo de desarrollo actual vs. Características del ciclo de desarrollo de MSF Ágil.

Antes de definir el modelo de ciclo de vida de desarrollo para un proyecto dentro del contexto señalado, hay que comprender los fundamentos básicos, los obstáculos y ventajas, esto incluye como primera medida, realizar un estudio de las prácticas que se van a poner en ejecución dentro de un proyecto. El modelo a aplicar en el proyecto debe ser estructurado teniendo en cuenta las características propias del proyecto. Un modelo de ciclo de vida exitoso en un contexto, no necesariamente lo es en otro contexto, en el caso del proyecto Juegos Consola se deben tener en cuenta las características propias del equipo, que sin altos recursos, necesita en poco tiempo el desarrollo de *software* altamente funcional y con la mejor calidad posible. Según la propuesta realizada en (Bauta, et al., 2007), muchas de las prácticas de MSF Ágil son las que más beneficios aportan al proyecto. En siguiente subepígrafe se parangonan las principales características del ciclo de desarrollo actual en el proyecto y las características del mismo, guiado por MSF Ágil.

2.1.3.1 Dirigido por casos de uso (CU) vs. Dirigido por escenarios y requerimientos de calidad de servicio (QoS).

Como en el ciclo de vida de RUP, el del proyecto Juegos Consola es dirigido por CU. Con esto se entiende que los CU guían el proceso de desarrollo, se obtienen modelos como resultado de los diferentes flujos de trabajo, que representan la realización de los CU (cómo se llevan a cabo). Los CU son el instrumento para validar la arquitectura del *software* y extraer los casos de prueba. Pero estos no brindan el nivel de especificación que requieren los integrantes del proyecto para el desarrollo de videojuegos, pues estos son muy complejos e ilustran muy poco, por tanto no permiten validar correctamente la arquitectura.

MSF intencionalmente distingue a los escenarios de los CU con diferentes términos porque defienden que los escenarios tienen un nivel mayor de especificidad, que no es capturada comúnmente en los CU. Los escenarios y los CU comienzan de forma similar pero se desenvuelven de manera diferente.

Para MSF Ágil, escenario significa el único camino de interacción de un usuario (o grupo de usuarios que colaboran representado por personas) que lo conduce a alcanzar sus objetivos establecidos, constituyen la principal forma de requerimientos funcionales. El objetivo que se persigue es la clave y es necesario que se exprese en el lenguaje del usuario.

A medida que se desenvuelve la solución los escenarios naturalmente se transforman en prototipos y funcionalidades operativas. Permiten al equipo planificar y medir el flujo de valor para el cliente. Posibilitan obtener una medición progresiva de la disposición y le da al equipo una magnitud de alcance. Todos los *stakeholders* tienen una visión común de los escenarios de manera que puedan ser expuestas las diferentes hipótesis, superar prejuicios y hacer tangibles las discusiones sobre los requerimientos.

Obtener una especificación temprana

Cuando las personas están definidas (el término "persona" para MSF Ágil es análogo al término "actor" para RUP), se pueden describir los escenarios para esas personas específicas. Se comienza con el objetivo del escenario. A medida que evoluciona la definición de la solución, se conformará el escenario en una secuencia específica de pasos con los datos obtenidos según el objetivo del escenario, éste en ningún momento debe cambiar. Entonces se divide el objetivo en una lista de pasos, a cualquier nivel de granularidad es más entendible el objetivo. Los pasos se enumeran con verbos de acción. Los escenarios deben mantenerse en el lenguaje del usuario, no en el lenguaje de implementación. En otras palabras, los escenarios deben describir desde el espacio del problema no desde el espacio de la solución.

Inicialmente no se deben detallar los flujos alternos y excepcionales, porque generalmente obstaculizan la comprensión y se tornan de manera rápida muy difíciles de controlar, pero sí debe quedar registrado su nombre como otro escenario para tenerlo en cuenta en futuras iteraciones.

Cuando la definición de la secuencia de pasos ofrece comodidad para la comprensión del escenario, se puede añadir la segunda parte para la definición, es decir, como responde a la solución deseada. Se hace una lista en el formulario, de los pasos de las personas y otra con los resultados de la solución.

Alcance de los escenarios

Es útil pensar en la historia de fin a fin que la solución necesita para describir y alinear los escenarios consecuentemente. Generalmente se deben tener los escenarios suficientes para definir los flujos principales de los objetivos del sistema y deben ser pocos para recordarlos fácilmente. Estos escenarios probablemente necesitan descomponerse en actividades intermedias, los cuales a su vez contienen los pasos. Si se crea un escenario multipersona, entonces es útil representar el flujo a través de las calles para mostrar el orden en que las personas proceden.

Requerimientos de calidad de servicio

Los escenarios deben ser comprendidos en el contexto de calidad de servicio. La mayoría de las insatisfacciones pueden ser eliminadas con una definición adecuada de calidad de servicio. Los QoS pueden definir atributos globales del sistema o puede definir limitaciones específicas en determinados escenarios. Por ejemplo: requerimientos de seguridad, "que los usuarios sin autorización no tienen permisos de acceder al sistema, ni a su información", es un QoS de seguridad global.

No todos los QoS se aplican a todos los sistemas, se debe saber cuáles se aplican según el mismo. Con frecuencia los QoS implican grandes riesgos o requerimientos de arquitectura, estos deben ser negociados con los *stakeholders* desde el comienzo del proyecto.

Se han producido varios estándares, éstos tienden a volverse obsoletos con el avance de la tecnología. Por ejemplo, la privacidad y la seguridad no se recogen en los principales estándares, a pesar de ser los más importantes en muchos sistemas modernos.

Algunos de los QoS más comunes a considerar en un proyecto:

- Seguridad y privacidad
- Rendimiento
- Experiencia de usuario
- Gestionabilidad

Como los escenarios, los QoS necesitan ser explicitados de manera entendible a sus *stakeholders*, definidos tempranamente, y si se planifican para una iteración deben ser comprobables.

Los escenarios y los QoS como *work ítems* que son, se almacenan en la base de datos común del VSTFS, ofrecen datos para la elaboración de informes, y permiten la asignación de tareas de manera ágil a través del VSTS.

2.1.3.2 Centrado en la arquitectura vs. Mayor administración de riesgos

La línea que sigue el proyecto Juegos Consola en el ciclo de desarrollo de videojuegos, es además centrada en la arquitectura como propone RUP; basándose en la confección de una visión común del

sistema completo, en la que todos los miembros del equipo, además de los usuarios, están de acuerdo. La arquitectura del proyecto describe los elementos del modelo que son más importantes para la construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. Este modelo de la arquitectura es representado a través de vistas en la que se incluyen los diagramas UML, se ha desarrollado la vista de casos de uso, la vista lógica, la vista de despliegue y además se utiliza el *framework* de la biblioteca *Graphics Three Dimensional* (G3D) para la implementación de la solución. Se elabora un plan de Contingencia de Riesgos, donde se registran los riesgos más elementales y se realiza un plan para mitigarlos, pero esto se hace en las etapas avanzadas del proyecto.

MSF Ágil propone un enfoque adaptativo, si el proyecto necesita responder rápidamente a la presión competitiva de negocio o la usabilidad, como es el caso de los videojuegos, no se puede diseñar perfectamente ni completamente desde el inicio, es por esto que se hace el diseño sobre la marcha del mismo, teniendo en cuenta nuevos requisitos.

El principal riesgo con el enfoque adaptativo es que se descubran cuestiones de diseño que obliguen a una revisión significativa o a cambios en el alcance del proyecto. Es por esto que MSF Ágil se enfoca principalmente en la mitigación de los riesgos en cada uno de los grupos de trabajo del modelo de equipo, con el fin de lograr una arquitectura robusta. La administración de riesgos es una disciplina básica dentro de MSF Ágil, reconoce que los cambios y la incertidumbre que estos generan son aspectos inherentes del ciclo de vida de un proyecto.

La disciplina de administración de riesgos de MSF Ágil prefiere tratar esta incertidumbre desde una perspectiva proactiva, realizando ininterrumpidamente valoraciones de riesgos que incidan en la toma de decisiones durante un ciclo de vida. Esta disciplina describe principios, conceptos y consejos, así como un proceso de cinco fases para realizar con éxito la administración continuada de riesgos: identificación de riesgos, análisis de riesgos, planeamiento de estrategias de contingencia y mitigación, control del estado de riesgos y aprendizaje de los resultados.

Las características que definen una administración de riesgos proactiva son la mitigación de riesgos y la reducción del impacto producido por los riesgos. La mitigación puede ponerse en práctica cuando el riesgo ya existe para intentar resolver la causa inmediatamente subyacente, o bien puede centrarse en la causa

raíz del problema (o en cualquier nivel de la cadena causal). Las medidas de mitigación se aplican mejor durante la fase preliminar de un proyecto porque el equipo todavía puede intervenir a tiempo para lograr el objetivo del proyecto.

Proceso de administración de riesgos de MSF Ágil.

Los seis pasos que conforman el proceso administración de riesgos de MSF son:

- Identificación
- Análisis y asignación de prioridades
- Planeamiento y programación
- Seguimiento y elaboración de informes
- Control
- Aprendizaje

Identificación de riesgos: Los riesgos se identifican y se ponen al descubierto. Esto debe realizarse lo antes posible y repetirse con frecuencia a lo largo de todo el ciclo de vida del proyecto.

Análisis y asignación de prioridades: Se transforman las cifras y los datos de los riesgos detectados en información que el equipo puede utilizar para tomar decisiones relacionadas con la asignación de prioridades. Al establecer la prioridad de los riesgos el equipo puede confirmar los recursos del proyecto para administrar los riesgos más importantes.

Planeamiento y programación de riesgos: Se toma la información obtenida tras el análisis de riesgos y se utiliza para trazar estrategias, planes y acciones. La programación de riesgos garantiza que estos planes se aprueban y se incorporan en la infraestructura. La programación de riesgos es el punto de conexión explícito entre el planeamiento de riesgos y el planeamiento de proyectos.

Seguimiento y elaboración de informes de los riesgos: Supervisa el estado de los riesgos y el progreso de sus planes de acción. El seguimiento de riesgos también incluye en la supervisión de probabilidades, impactos, exposiciones y otras medidas de riesgo para los cambios que pudiesen alterar los planes de prioridades o de riesgos y las características, los recursos o la programación del proyecto.

El informe de los riesgos garantiza que el equipo esté al corriente del estado de los riesgos del proyecto y de los planes para administrarlos.

Control de riesgos: Es el proceso que ejecuta los planes de acción de riesgos y sus informes de estado asociados. Incluye la iniciación de las solicitudes de control de cambios del proyecto si los cambios en el estado o los planes de los riesgos pueden alterar las características, los recursos o la programación del proyecto.

Aprendizaje de riesgos: Formaliza las lecciones aprendidas, los elementos y herramientas relevantes del proyecto y plasma toda esta información en un formato reutilizable para el equipo o la empresa.

Gestión de riesgos

Los riesgos que enfrentan los proyectos difieren considerablemente. La gestión de riesgos tradicional tiende a ser guiada por eventos, de la forma "¿Qué pasa si ocurre X?". MSF complementa la gestión de riesgos guiada por eventos con un modelo basado en áreas**,** que incluye siete puntos de vistas que necesitan ser representados en un proyecto para prevenir puntos ciegos.

En algunos casos, por ejemplo, la tecnología es nueva y necesita ser probada antes de ser adoptada. A veces cuestiones extremas de los QoS deben ser abordados, como la escalabilidad o el rendimiento. Otras cuestiones significativas que influyen son la creación de equipos nuevos que desconocen las habilidades y estilos de comunicación. VSTS, a su vez, da seguimiento a los riesgos del proyecto directamente en la base de datos de *work ítems* con el objetivo de que el equipo de desarrollo los tenga a su disposición directamente.

Puntos de vista de los Riesgos: Intereses de los Grupos

Descripción de los Intereses de los Grupos en el Modelo de Equipo de MSF

La administración del proyecto aboga por la entrega de la solución:

El objetivo de la administración del proyecto es responsabilizarse con la entrega del producto de *software* cumpliendo con las restricciones del cliente. Asegura la entrega en tiempo de una solución que cumpla

con todas las expectativas y requerimientos que se acordaron entre los *stakeholders* y los desarrolladores durante la concepción y evolución del proyecto.

La arquitectura aboga por un sistema extensible:

Incluye los servicios, la tecnología y estándares con los cuales la solución interoperará, la infraestructura en la cual ésta será desplegada, los usuarios a los que está dirigido, así como el soporte y la extensibilidad para futuras versiones. El grupo de arquitectura debe asegurar que la solución entregada cumpla con todas las cualidades del servicio, con los objetivos del negocio y que sea viable.

El desarrollo aboga por la Solución Técnica:

Además de ser la solución primaria de los desarrolladores, el desarrollo es confiable para las decisiones técnicas significativas, para un diseño claro, un código sostenible y de calidad y para las pruebas unitarias.

Las pruebas abogan por la calidad de la solución desde la perspectiva del cliente:

Las pruebas anticipan, buscan, y reportan cualquier cuestión que pueda disminuir la calidad de la solución desde la perspectiva de los clientes o usuarios.

El release aboga por una solución refinada y por su despliegue en una infraestructura apropiada:

Este grupo asegura la disponibilidad de la solución y su compatibilidad con la infraestructura final.

La experiencia del usuario aboga por una solución más efectiva desde la perspectiva de los usuarios finales.

La experiencia del usuario debe entender el contexto del usuario como un todo, estimar cualquier sutileza de sus necesidades, y asegurar que todo el equipo esté consciente de la usabilidad desde su perspectiva.

La administración del producto aboga por el negocio del cliente:

La administración del producto tiene que comprender, comunicar, y asegurar el éxito desde el punto de vista del cliente económico que solicita el producto.

2.1.3.3 Modelo Incremental vs. Modelo Iterativo e Incremental.

En el proyecto Juegos Consola el ciclo de desarrollo se ha llevado a cabo utilizando un modelo incremental que consiste en aplicar el modelo tipo cascada, el cual origina una primera versión con su respectiva funcionalidad. Este proceso se aplica cada vez que el proyecto desea crear una versión más completa según los requerimientos del cliente. Este modelo se aplica en el proyecto porque tiene establecido un límite de tiempo para obtener la solución.

El Modelo en Cascada, es sencillo ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el *software*. Está basado en el ciclo convencional de una ingeniería, el paradigma del ciclo de vida abarca las siguientes actividades:

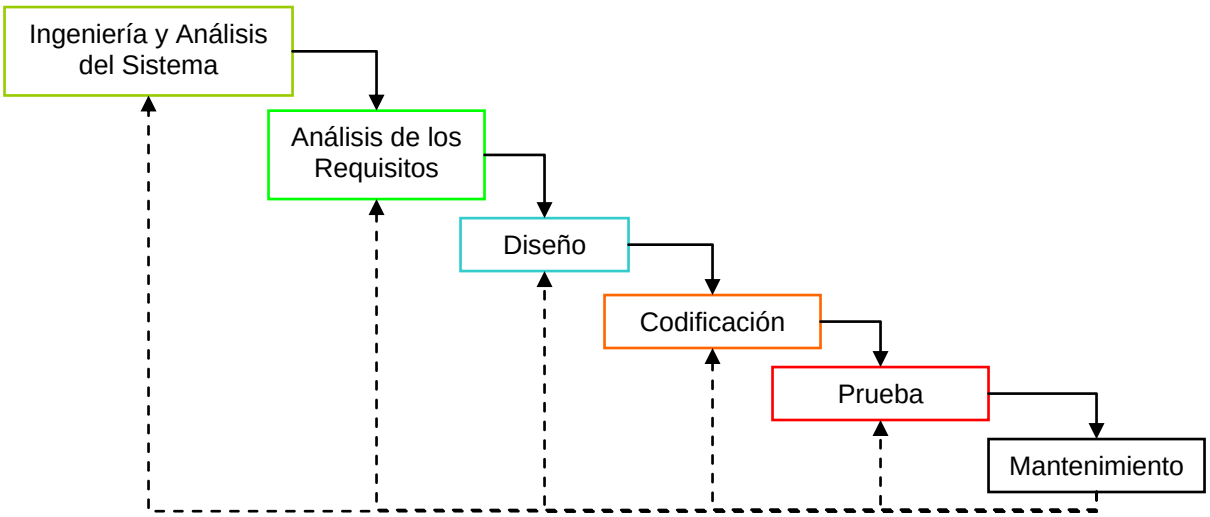

Fig. 2.2 Actividades del ciclo de vida del Modelo en Cascada

En el proyecto hasta el final no se dispone de una versión operativa del programa, pues la integración no se hace hasta entonces. En varias ocasiones no se detectan errores importantes hasta que el programa está funcionando, es cuando resulta un verdadero desastre.Las pruebas comienzan a efectuarse luego de haber terminado la implementación, lo que trae como consecuencia un "*roll-back*" de todo un proceso que costó tiempo y recursos.

Una de las características del proyecto es que resulta difícil para el cliente establecer explícitamente desde el principio todos los requisitos, y una de las desventajas de este modelo es que tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.

En cambio MSF Ágil propone que el desarrollo sea iterativo e incremental, al aplicar al modelo iterativo no se le agrega funcionalidad si no que en cada iteración se mejora la misma. Permite crear cada vez versiones más completas de *software*, para esto se construyen versiones sucesivas de un producto.

Los ciclos describen la frecuencia con que se realizan las actividades. Estos siguen la ejecución del proyecto y de sus tareas. Dentro de los eventos del ciclo de vida existe una jerarquía de proceso denominada iteración de proyecto. Al crearse por vez primera el proyecto de equipo, su ciclo de vida se expresa en términos de iteraciones (período fijo de tiempo en el cual programar tareas). Éstas se numeran de forma consecutiva y se organizan de manera continua. Este desarrollo iterativo se presenta como antídoto al desarrollo secuencial en “cascada”.

La integración continua de MSF Ágil para el desarrollo ágil de *software* en *Visual Studio Team System* contribuye al desarrollo iterativo ágil, en perfeccionamiento y continuo aprendizaje. La definición del producto, el desarrollo, y las pruebas se producen en la superposición de iteraciones resultantes en incremental completamiento del proyecto. Las distintas iteraciones tienen diferentes enfoques como los *release* del proyecto. Las iteraciones pequeñas permiten reducir el margen de error en las estimaciones y proporcionar rápidamente información acerca de la precisión de los planes del proyecto. En cada iteración se debe obtener una parte estable del sistema.

Al principio de una iteración se planifica el trabajo que se va a abordar a corto plazo, y al final de ésta se arriba a conclusiones del desarrollo de la misma sobre los resultados obtenidos. Este proceso de “mirar atrás” tiene como objetivo fundamental avanzar positivamente. Pues según los resultados de la iteración anterior se obtiene una guía para planificar la siguiente iteración.

El proceso descrito se puede aplicar de manera anidada. Una iteración puede contener a otras. Esto es así porque al principio de un proyecto es habitual no tener un alto nivel de detalle sobre cuáles serán las iteraciones, y por lo tanto se pueden definir 3 ó 4 iteraciones que luego se irán refinando en sub-iteraciones.

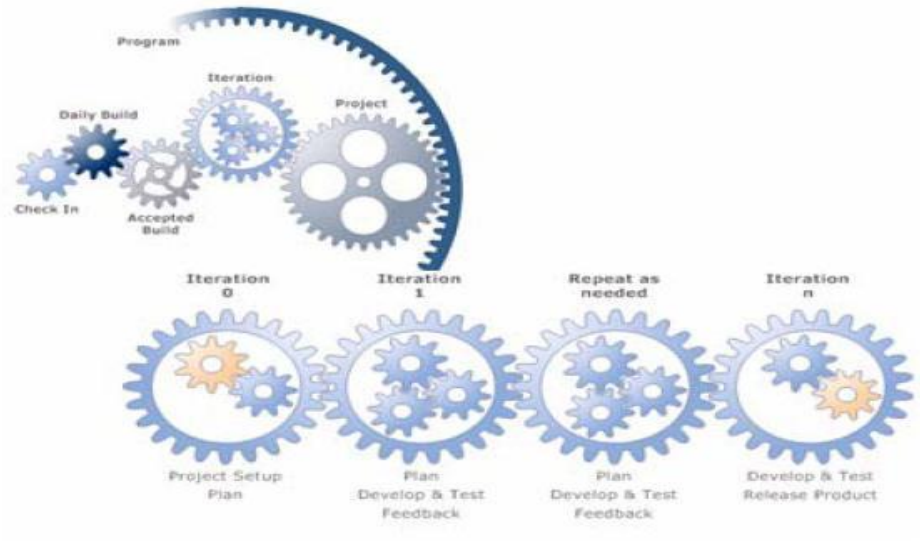

Fig. 2.3 Ciclos e Iteraciones (Medela, 2006)

A la hora de construir iteraciones se debe establecer una meta. La duración de las iteraciones es fija en algunas metodologías, pero en MSF Ágil por ejemplo, se decide como parte de la planificación de la misma.

Teniendo en cuenta lo logrado en la iteración anterior, es necesario en la planificación de una iteración:

- Decidir cuánto durará
- Establecer qué trabajos se realizarán
- Estimar en detalle los trabajos
- Asignar quién realizará los mismos
- Establecer una meta para la iteración

Usando iteraciones, el intervalo en el cual se pensaron escenarios mide el flujo del valor, determinan el proceso real, examinan embotellamientos y hacen ajustes.

De esta forma las tareas de desarrollo, se organizan separadamente en iteraciones, con el fin de que en cada una de ellas, se logre una pieza de *software* construida sobre la versión lograda en iteraciones previas. Conjuga el concepto iterativo, con la idea de aceptar los cambios en los requerimientos. Permite la resolución de problemas de alto riesgo en tiempos tempranos del proyecto. Se obtiene una visión de

avance en el desarrollo desde las etapas iniciales del desarrollo y un *feedback* del usuario lo antes posible, para orientar el desarrollo al cumplimiento de sus necesidades y realizar todas las adaptaciones identificadas para cumplir con los objetivos planteados.

Favorece una menor tasa de fallo del proyecto, mejor productividad del equipo, y menor cantidad de defectos. Se maneja la complejidad del proyecto, apuntando a la resolución de los problemas por partes, y no cae en la inanición del "súper análisis" del producto. El aprendizaje y experiencia del equipo iteración tras iteración, mejora exponencialmente el trabajo, aumenta la productividad y permite optimizar el proceso en el corto plazo.

El trabajo iterativo deja una experiencia en el equipo que permite ir ajustando y mejorando las planificaciones, logrando menores desvíos en la duración total del proyecto. Además, su adopción no presenta grandes inversiones.

Los jefes del proyecto deben tener una vista total del trabajo que harán, de la capacidad de supervisar y de ceder prioridad con frecuencia en los límites cortos de la iteración. Esta puesta al día frecuente requiere una reserva fácilmente visible, de preferencia con la colección de datos automatizada, tal como la base de datos de *work ítems* que VSTS proporciona.

2.1.3.4 Métricas Prescriptivas vs. Descriptivas

Actualmente en el proyecto Juegos Consola las métricas son analizadas de manera prescriptiva y aisladas, buscando una manera de estar en ciertos rangos. Utilizar las métricas prescriptivas para evaluar puede llevar a generar una cultura de cumplimiento, en la que se pone el proceso sobre las personas, y se olvida su importancia. Otro factor dañino de utilizar las métricas para evaluar, es que los desarrolladores se empeñan en mejorar esas métricas sobre las que se evalúa, olvidando aquellos aspectos del proceso de desarrollo que no se evalúan y que siempre existen.

Más que necesarias para evaluar, las métricas constituyen un arma de suma importancia para ganar información sobre los proyectos. En este epígrafe se defiende la superioridad de usar métricas de una manera descriptiva, viéndolas de una manera gráfica y multidimensional, y relacionadas entre ellas. La mirada descriptiva, multidimensional y gráfica a las métricas, no es constante, varía de proyecto a proyecto e incluso en diferentes momentos de un mismo proyecto.

Esta visión de las métricas es la que ayuda a mejorar el proceso de desarrollo. No es tan cómoda como la visión prescriptiva, pero es mucho más ágil y permite detectar problemas mucho antes en aspectos sobre los que no se tienen métricas explícitamente establecidas. El acercamiento a las métricas de manera descriptiva, multidimensional y gráfica es el que propone MSF e implementa VSTS.

Almacén de Métricas.

VSTS provee e instrumenta el proceso, vinculando el código fuente, las pruebas, los *work ítems* y las métricas.

Los *work ítems* incluyen todo el trabajo que necesita ser rastreado en un proyecto. Éstos pueden ser detectados y corregidos en el *Team Explorer*, *Visual Studio, Microsoft Excel* o *Microsoft Project*.

En una base de datos común, un conjunto de herramientas descifran la información. En lugar de cortar y pegar entre artefactos distribuidos aleatoriamente, el jefe de proyecto, analista de negocio, desarrollador y probador ven el mismo trabajo, a diferencia de otros proyectos que utilizan herramientas y técnicas de seguimiento separadas, gran parte de la recopilación de datos en VSTS es automática.

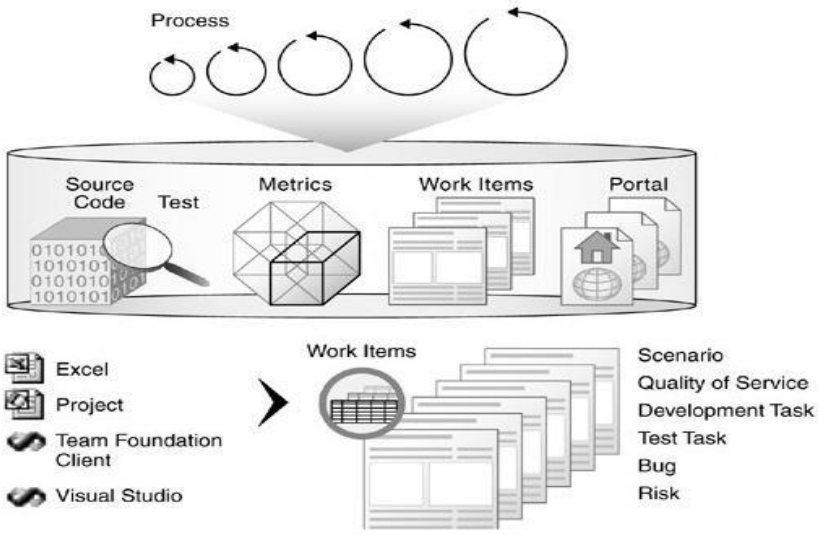

Fig. 2.4 Base de datos común del TFS (Guckenheimer, 2006)

Reunir datos automáticamente, correlacionándolos, y usándolos para medir el proceso.

VSTS toma este enfoque. Instrumenta las actividades diarias de los miembros del equipo para recoger datos del proceso sin sobrecarga. Por ejemplo, chequea el código actualizado dentro del control de versiones, los *work ítems* son actualizados para reflejar las tareas y escenarios actualizados por éste código. En las conexiones son capturados un conjunto de cambios y cuando se ejecuta el próximo *build* éste identifica el conjunto de cambios y actualiza los *work ítems* nuevamente con el número de *build*. Cuando se ejecutan las pruebas, se usa el mismo número de *build*. Entonces los resultados de las pruebas, los cambios en el código y los *work ítems* son relacionados automáticamente por el *VSTS.*

El almacén de métricas reúne los datos de todas las acciones en el proyecto para proveer los reportes que relacionan las diferentes fuentes y dimensiones de los datos.

Además, para mantener la acumulación de datos actualizados y visibles, este colector automático de datos permite a través del almacén de datos de métricas revelar las tendencias y comparaciones de calidad día por día desde varias dimensiones. De esta forma el almacén de datos brinda inteligencia al negocio en funciones como ventas o procesos de producción, éste provee inteligencia en el proceso de desarrollo de *software*.

Con este almacén de datos, preguntas básicas sobre el progreso y el estado en tiempo real del proyecto se hacen fáciles de responder, los jefes de proyecto deben preocuparse por responderlas con datos precisos. Cuando se automatiza la colección de datos, las respuestas son convenientemente directas.

Métricas multidimensionales

La capacidad de ver varias dimensiones de los datos del proyecto es un beneficio directo del almacén de métricas, el cual reúne y relaciona los datos de las actividades diarias. De esta forma se puede lograr el nivel de visibilidad necesario para el reporteo de estricto cumplimiento mientras se trabaja de manera ágil y tener la misma visibilidad en un proyecto remoto, incluso como si se tuviera en el mismo local.

Varias dimensiones de la "salud" de un proyecto

Los proyectos de *software* tienen muchas dimensiones que interactúan, y muchas de ellas pueden ser relevantes. Observar estas dimensiones ofrece una oportunidad para el descubrimiento en tiempo de excepciones y embotellamientos que necesitan correcciones. Las dimensiones múltiples ayudan a asegurarse de que las mediciones siguen un reporte coherente.

Entre los reportes más comunes que deberían usarse para monitorear un proyecto se incluyen:

- Tiempo dedicado a las tareas
- Conteo de *Bugs*
- Completamiento de las tareas, según los distintos reportes
- Utilización de recursos en términos de contabilidad

Cada uno de ellos suele medirse por separado.

A diferencia de los viejos modelos de proyecto, donde las métricas del proyecto se obtienen de manera aislada, VSTS combina las mediciones en el almacén de métricas usando un conjunto de cubos de análisis y reportes en las dimensiones conjuntamente.

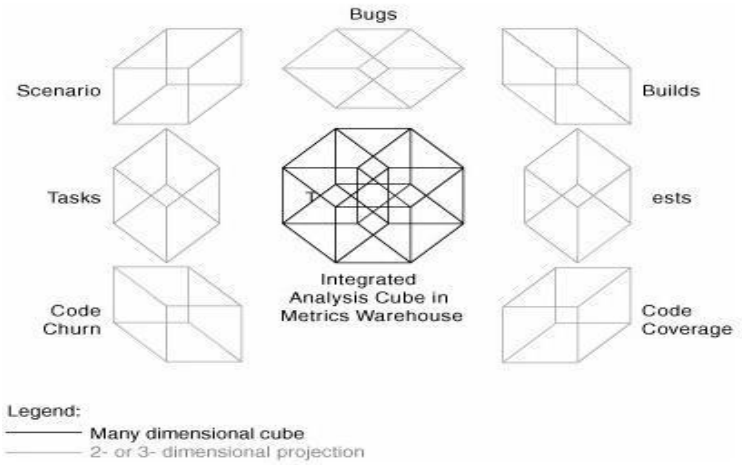

Fig. 2.5 Almacén de Métricas (Guckenheimer, 2006)

Se puede hacer el mismo análisis en todos los *work ítems* independientemente del tipo que sean. Los cambios en los *work ítems* se convierten en métricas que automáticamente siguen el progreso para proporcionar datos para la próxima estimación. La calidad puede ser medida contra el avance de las tareas usando varios puntos de vista a la vez.

2.1.3.5 Pruebas actuales vs. Pruebas requeridas.

En el proyecto Juegos Consola en el trabajo actual se han llevado a cabo únicamente por parte del probador, las pruebas de funcionalidad para comprobar las funciones luego de la integración, además pruebas de usabilidad, de rendimiento, y la de soportabilidad que se realiza con el objetivo de comprobar que el juego se pueda ejecutar en cualquier PC con los requerimientos adecuados. La no realización de pruebas unitarias puede ocasionar retrasos en el avance del proyecto, éstas cobran gran importancia al ser las más exhaustivas, ya que implica hacer pruebas por clase, función, u otras unidades, hasta llegar a la que se considere mínima. Permite probar trozos de código sin implicar al resto. Brinda una mayor confianza para modificar código, por lo que se pierde el miedo al hacer modificaciones y volver inconsistente el sistema. Además permite encontrar fallos en la interfaz de la clase en pasos prematuros, evitando así el coste que supone una modificación en fases más avanzadas.

Hay que tener en cuenta lo que se plantea en (Bauta, et al., 2007) que aunque el VSTS trae de forma integrada un *framework* de prueba, muchos desarrolladores optan por usar *NUnit* en lugar del propuesto por *Microsoft*. *NUnit* surge bajo la copia fiel del *framework* de prueba *JUnit* que existía para aplicaciones Java. Es un marco para desarrollar pruebas unitarias automatizadas que permite que el código se compile y que las pruebas se ejecuten frecuentemente. Esta herramienta además de ser *Open Source*, facilita la realización de éstas pruebas que tanta importancia presentan para el desarrollo de videojuegos. Si el código a usar no es 100% confiable, no debe ser utilizado en la solución, de ahí la necesidad de realizar este tipo de pruebas, ya que producir calidad en el código está directamente relacionado con la cantidad y la minuciosidad de la prueba realizada contra ese código.

Según la recomendación realizada en (Bauta, et al., 2007) poner en práctica en el proyecto de Juegos Consola la versión *Suite* del *Visual Studio,* por la oportunidad que brinda al equipo de desarrollo, de establecer la comunicación y colaboración ideal. Además combina las funcionalidades de las cuatro ediciones basadas en funciones de VSTS (*Team Editions* para arquitectos, desarrolladores, profesionales

de base de datos y probadores) con el fin de ofrecer un entorno de desarrollo completo a los miembros de equipos de varias disciplinas. En el presente epígrafe se describen las diferentes ventajas para el desarrollo de las pruebas en el proyecto, con la adopción de MSF para el desarrollo ágil y su integración a la herramienta para la realización de las pruebas.

Ahora es posible administrar el ciclo de vida de prueba**,** dada la plena integración con TFS, que ofrece una vista sencilla de las métricas de las pruebas, los códigos y los requisitos. Además MSF Ágil posibilita que las pruebas se puedan administrar a lo largo del ciclo de vida de desarrollo de *software* del mismo modo que el código. Y la interfaz de usuario, las herramientas y los lenguajes de código de gran simplicidad permiten a los desarrolladores y personal de pruebas colaborar de forma más eficaz.

Las técnicas y herramientas de prueba se distribuyen según el rol.

Los **probadores** realizan las pruebas:

Prueba de carga: Simula múltiples usuarios ejecutando las pruebas automatizadas.

Prueba Web: Simula cómo los usuarios interactúan con una aplicación Web.

Prueba Manual: Se realizan y se registran en un documento las pruebas no automatizadas.

Administración para casos de prueba: Es usado por los probadores para categorizar las pruebas.

Mientras que los **desarrolladores** las:

Prueba Unitaria: Código que ejercita los métodos del proyecto.

Prueba de Cobertura: Asegura que las pruebas unitarias cubren todas las posibles rutas o caminos del código.

Análisis Estático: Asegura que el código sigue los lineamientos de codificación establecidos, mediante la inspección de binarios.

***Code Profiling*:** Desempeño basado en la instrumentación de binarios.

Según (Guckenheimer, 2006), las pruebas además de cubrir los requerimientos identificados, o lo que es lo mismo los escenarios conocidos, para considerarse buenas deben descubrir otros nuevos. Una de las afirmaciones del paradigma *value-up* expresa que los conocimientos aumentan a medida que avanza el proyecto, y con frecuencia surgen a partir de los descubrimientos. Se deben identificar y capturar los nuevos escenarios que se encuentren durante la planeación y ejecución de las pruebas.

Las pruebas se especializan en función de los diferentes QoS que son capturados en la base de datos de *work ítems*. Por ejemplo, las pruebas de carga, las pruebas de configuración, las pruebas de seguridad y las pruebas de usabilidad son radicalmente diferentes.

Prueba unitaria

Consiste en un código que ejercita los métodos del proyecto, se crea un método en un proyecto de prueba al cual se le asignan los valores a los parámetros de entrada y a los de salida. Cuando se ejecuta la prueba, se evalúa el resultado previsto con el generado en tiempo de ejecución, presentando el fallo o éxito del método de prueba. Y el orden de creación de los métodos es indistinto, es posible crear los "métodos de prueba" primero que los "métodos a desarrollar", o ambos interactivamente.

Cobertura (cubrimiento de código)

Esta técnica asegura que las pruebas unitarias cubran todas las posibles rutas o caminos del código. Como resultado brinda un porcentaje de código cubierto por las pruebas así como las líneas cubiertas. Por tanto es necesario tener pruebas unitarias y que el código esté corregido. Además ésta evalúa si una línea de código está cubierta y le asigna un color. Color Verde a la línea leída, Naranja a los predicados analizados incompletamente y rojo a las líneas no evaluadas o leídas por la prueba/s.

Genéricas

Es el tipo de prueba que encapsula una prueba o herramienta desconocida y permite a *Visual Studio* tratarla como si fuera de tipo conocido. Esta aplicación se debe de poder ejecutar desde línea de comando y debe retornar un valor de "Sin errores" (*return* 0) o "*Error*" (*return* <>0).

Ordenadas

Ejecuta un conjunto de pruebas existentes en un orden determinado (ej. *UTest* en orden específico). Puede tener todos los tipos de pruebas menos las de carga, las pruebas manuales que se ejecutan remotamente o por línea de comando que no se ejecutan indicando un *warning* en la pantalla. Éstas se pueden ejecutar desde el IDE y/o línea de comando.

Análisis estático

Asegura que el código siga los lineamientos de codificación establecidos. Además diagnostica errores críticos de seguridad y desempeño mediante la inspección de los binarios sin necesidad de ejecutar el código en producción.

Prueba de carga

El propósito principal para los probadores es simular múltiples accesos de usuarios (alrededor de 1000 por procesador) al mismo tiempo a un servidor. Prueban al sistema en condiciones extremas permitiendo evaluar el desempeño, rendimiento y tiempos de respuestas. Las pruebas de carga se pueden dividir bajo dos modalidades agregando pruebas *web* o pruebas unitarias. Para el primero simula apertura de varias conexiones realizando solicitudes http, con propiedades *web* (versiones de navegadores, anchos de banda, contadores). La segunda ensaya el rendimiento de componentes del servidor no basados en *web* (ej. modelo de acceso a datos).

Tipos de pruebas de carga que existen:

- **Humo:** Cómo se comporta su aplicación bajo cargas ligeras de duración corta.
- **Tensión:** Para determinar si la aplicación puede ejecutarse correctamente de forma sostenida bajo una carga intensa.
- **Rendimiento:** Para determinar cómo responderá su aplicación.
- **Diseño de la capacidad:** Cómo se comportará su aplicación a distintas capacidades.

VSTS recoge automáticamente los datos del diagnóstico de los servidores bajo pruebas para poner a relieve los problemas existentes. En VSTS una prueba de carga es un contenedor para cualquier grupo de pruebas con la configuración de los elementos de carga de trabajo. A menudo se desea el sistema con un

incremento gradual de la carga del usuario para poder detectar cualquier efecto negativo en el tiempo de respuesta según aumenta la carga del mismo. Luego se selecciona la prueba (unitaria, *web*, u otra) y el porcentaje de carga para crear cada una de las pruebas atómicas. El próximo paso es seleccionar el navegador y las combinaciones de red que mejor reflejan la población del usuario final.

Las pruebas de carga pueden generar grandes cantidades de datos de los servidores bajo prueba, y a menudo es difícil saber qué es lo relevante. VSTS simplifica esto porque permite que se seleccionen sólo aquellos servicios a observar y en las máquinas correspondientes.

Pruebas Manuales

Es un conjunto de pruebas que se realizan en forma manual, ya que son complicadas hacerlas de otra manera, o no se podrían hacer. Las pruebas están formadas por un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, desarrolladas para un objetivo concreto. El probador las genera documentando paso a paso en un documento *word* que está vinculado dentro del *Solution Explorer*, luego el evaluador ejecuta los pasos y se completan los resultados en el *Visual Studio*. Al momento de ejecutar la prueba en el *Visual Studio* le pregunta al usuario si la prueba pasó o no, adjuntando un comentario si es necesario.
En VSTS se pueden describir las pruebas manuales en un documento. Los resultados de las pruebas son capturados, rastreados, y proveen al almacén de la misma forma que las pruebas automatizadas.

2.1.4 *Visual SourceSafe* vs. Control de Versiones con VSTS

En el proyecto Juegos Consola para el control de versiones se utiliza el *Microsoft Visual SourceSafe* 6.0. La función principal de *Visual SourceSafe* consiste en realizar un seguimiento de los historiales de los archivos y proyectos de la base de datos. Un historial incluye todas las versiones de un archivo o proyecto, desde la versión inicial a la actual, de la base de datos. Se puede ver el historial de un archivo o proyecto y recuperar cualquier versión del elemento desde la base de datos.

Esta herramienta es considerada muy práctica y fácil de usar, con frecuencia es la principal herramienta considerada para administración de la configuración usada por desarrolladores, pero desafortunadamente no es una herramienta que cumpla con todos los requerimientos de una herramienta de administración de la configuración y la desventaja más grande es que no está enfocada para equipos de desarrollo grandes

si no para desarrolladores únicos o bien equipos de desarrollo muy pequeños, lo que significa que no es escalable y puede no ser ventajoso su futuro uso en el proyecto, el método de almacenamiento es por medio de sistemas de archivos el cual es una gran desventaja, el control de versiones será en dependencia de cada punto de vista.

En cambio el control de código fuente del VSTS se realiza a través del *Team Fundation Version Control* (TFVC), es una aplicación basada en tres capas y su arquitectura está basada en *Web Services*, su almacenamiento es en SQL *Server* para lograr un control de versiones transaccional, es escalable según el tamaño del equipo, cuenta con integraciones con otras herramientas de VSTS para la administración de automática de flujos de trabajos, implementación de conceptos avanzados de administración de la configuración basados en CMMI. Soporta migraciones desde *Visual SourceSafe, Rational ClearCase*.

	Visual SourceSafe 2005	Visual Studio 2005 Team Foundation
Tareas	Versión de Control Simple	Software Integrado al Ciclo de Vida
Tamaño del Equipo	Individual, equipos pequeños	Escalable
Almacenamiento	Sistema de Archivos	SQL Server 2005
Seguridad	Aplicación Especifica	Integrada con Windows
Acceso Remoto	Nuevo Web Service para Integración con VS	Optimizado para Web Service

Tabla 2.1 Ventajas del TF para el control de código fuente

Los puntos favorables de la herramienta son principalmente los siguientes:

- Escalabilidad
- Fiabilidad
- La posibilidad del desarrollo de forma remota, mediante SOAP/http(s).
- Desarrollo en paralelo. Múltiples *releases*, bifurcaciones, combinaciones, múltiples *check outs* de un mismo archivo y luego puede combinarse.
- Experiencia de integración con *check in*. La combinación *source change*, comentarios, *work ítems*, políticas y notificaciones, contribuyen integrando y ajustándose al proceso del proyecto.

Las políticas de *check in* permiten disparar *Builds*, Pruebas unitarias, Análisis estático, pudiendo customizar y realizar extensiones que se ejecutan con el mismo. También permiten la asociación con los *work ítems*, y con notas de protección de los cambios introducidos al hacer el *check in*.

Checking In

VSTS incorpora en el control de versiones todos los ficheros abiertos relacionados en la solución del *Visual Studio*, y cualquier otro que se quiera incluir. Cuando se comienza a editar un fichero, el desarrollador se apropia de éste, a menos que alguien más lo tenga bloqueado (por defecto, una sola persona puede editar un mismo fichero a la vez, pero se pueden habilitar múltiples pedidos en el menú del *Team System*).

Conjuntos de cambios

La forma principal en la que se interactúa con el control de versiones es a través de la actualización de los ficheros. Cuando se actualiza, el fichero está listo para ser usado en el *team build*. VSTS tiene en cuenta tres cosas: la lista de ficheros, los *work ítems* que se resuelven con este *Check-in*, y las notas del *Check-in,* que son usadas para describir los cambios. En conjunto, estos tres elementos, integran el conjunto de cambios, los cuales relacionan los datos de un *check-in*. El conjunto de cambios incluye las líneas de código adicionadas recientemente, las modificadas o las eliminadas, los cambios de estado de los *work ítems* asociados a ese código, y las notas del *check-in*.

Un importante beneficio de los conjuntos de cambios en VSTS es que el *build del* sistema, al almacén de métricas rastrear adecuadamente por separado, la relación de los cambios de *work ítems* con el código fuente y probar los cambios. El *build* puede identificar los conjuntos de cambios que han sido enviados a la solución y por tanto identifica todas las transiciones de *work ítems* y calcula el código de *churn*. Reportes tales como el Reporte *Build*, y los Indicadores de Calidad, se basan en la estructura del conjunto de cambios.

Espacio de Trabajo

Es un espacio aislado, similar al directorio de trabajo del VSS, donde se puede escribir y probar el código fuente sin tener que preocuparse sobre cómo pueden afectar las modificaciones a la estabilidad del código

fuente protegido o cómo puede verse afectado por los cambios que realizan los demás miembros del equipo. Los cambios pendientes o "*pending Changes*".se aíslan en un espacio de trabajo hasta que sean protegidos en el servidor de control de código fuente. También se puede mapear un espacio de trabajo del TFS a un directorio local, para trabajar de forma desconectada realizar los cambios "sin control" y luego resolver los conflictos.

Bifurcar

Consiste en separar, una parte del código con el fin de probarla en forma local y luego combinarlas. Tener varias ramas permite mantener versiones paralelas de ficheros que se desarrollan individualmente. Probablemente el uso más frecuente de las ramas es rastrear múltiples *release* de una solución. Cuando se libera la primera versión, se puede ramificar para comenzar el trabajo en la segunda versión. Si se necesita posteriormente corregir errores o emitir un servicio de liberación (tal vez para los nuevos entornos) para la primera versión, se puede hacer en su rama sin tener que trabajar en el código de la segunda versión.

El uso de las ramas debe ser moderado. Cada vez que se crea una rama, se puede estar creando una necesidad futura de fusión. Los errores que se corrigen para la primera versión probablemente necesitan ser corregidos también en la segunda versión. Si se tienen varias ramas se tendrán varias combinaciones a considerar.

Aplazando

Permite hacer una versión especial con el espacio de trabajo en determinado momento, el cual se podría utilizar para compartir, realizar una copia de seguridad, o resetear un espacio de trabajo con uno archivado, es muy útil para un número de casos. Si se necesita salir del proyecto cuando el código no está listo para la compilación, se pueden realizar salvas. Si se necesita compartir el código con alguien antes de actualizar, para una revisión de código o una prueba, se puede separar el código y que alguien lo revise a partir del conjunto de cambios aplazados. Cuando se quiere experimentar con dos soluciones al problema, se puede tratar una, por separado, tratar la segunda e intercambiar entre los conjuntos de cambios aplazados para la prueba local.

Políticas de *Check-in*

Cuando se actualiza, VSTS también verifica que se haya cumplido con las políticas de *check-in* del equipo. Tres políticas estándares de *check-in* aseguran que se tengan *work ítems* asociados con los cambios, que se hayan llevado a cabo pruebas unitarias, y que se haya realizado el análisis de código estático. El equipo puede establecer otras políticas y también evaluarlas en el *check-in*.

Reporte del *Build*

Se muestran los resultados del completamiento del *build*, del análisis del código y de la ejecución de las pruebas de comprobación de *build* (BVT, por sus siglas en inglés). En el caso de una compilación fallida, se proporciona un vínculo al *work ítem* que fue creado para informarlo al miembro del equipo que corresponde, para que éste lo corrija y reinicie la compilación.

El reporte se utiliza para monitorear la ejecución del *build*, mostrar los detalles y los cambios de *work ítems,* documentando los sucesos del *build*. Abriendo los detalles de los resultados de las pruebas, mostrando los resultados de las BVT y el alcance del código de las BVTs. Los *warning* de compilación incluyen los resultados del análisis del código estático. Cuando falla un *build* un *work ítems* es creado automáticamente para rastrear el fallo, y éste también es mostrado. Dicho reporte permite publicar el estado de la calidad del *build*.

Integración Continua

La integración continua se refiere a la práctica de desencadenar una compilación después de cada *check-in*. Ha demostrado un gran éxito, porque retroalimenta con información de manera inmediata sobre los errores de integración para el desarrollador que acaba de actualizar. Con VSTS se puede establecer un tipo de compilación para la integración continua y desencadenar la compilación desde los eventos de check-in. Cuando se hace esto, se usa un tipo de compilación separada y distinta a la diaria, que mantiene la existencia de métricas de la compilación diaria.

Diferencias entre TFS SC y VSS

En TFS existen grandes cambios con respecto a los conceptos de *Check In* y *Check Out* de archivos, en relación al VSS. Pues maneja el concepto de espacio de trabajo, el servidor está consciente en todo momento del trabajo realizado en el espacio de trabajo, conoce los directorios locales que se utilizan para el mismo, las versiones de archivos que existen y los cambios pendientes que se han informado al servidor. Para conseguir la última versión de un archivo, a diferencia del VSS, se debe llamar explícitamente a *Get Latest Version*.

Cuando se hace un *Check Out* de un archivo, en realidad se le dice al server que se está por editar una versión específica de ese archivo; en este momento se pueden elegir una de las opciones siguientes:

- ***Lock None:*** el archivo no se bloquea, alguien dentro del equipo de trabajo puede editar el mismo archivo y subir los cambios al servidor. En el momento de subir los cambios locales, se deberá resolver el conflicto de versiones si es necesario.
- ***Lock check out:*** El archivo se bloquea en el servidor, y ninguna otra persona dentro del equipo tiene acceso a modificar este archivo (igual que en VSS).
- ***Lock Check In:*** el archivo no se bloquea en el servidor, pero no se permiten modificaciones al mismo, hasta que la persona que ha hecho el ***Lock Check In***, suba su versión.

En cambio cuando se hace in ***Check In*** de un archivo, los cambios son más completos; se puede hacer obligatorio que antes de subir un archivo, se ejecute el análisis del código y se realicen pruebas unitarias, además de asociar cada una de estas subidas, a una tarea previamente asociada al plan de trabajo y de esta manera controlar los cambios que implica cada tarea.

Además TFS permite el ***Shelving***; opción que posibilita “subir” archivos al TFS sin la necesidad de marcarlos, en estado ***Check In***, de esta manera se puede subir el trabajo diario al repositorio de archivos, sin que forme parte de las compilaciones diarias

CAPÍTULO III: PERSONALIZACIÓN DE LA PLANTILLA DE PROCESOS MSF PARA EL DESARROLLO ÁGIL.

Introducción

Los procesos de desarrollo de *software* son complejos, incluyen muchos niveles de actividades interdependientes. Estos procesos están disponibles para el equipo de desarrollo como documentación, pero realmente no son aplicados por herramientas. La falta de apoyo de éstas dificulta a los equipos de desarrollo el aprendizaje y el seguimiento de los procesos de manera coherente. Los jefes de proyecto pueden usar una gran variedad de herramientas para la gestión de proyectos, gestión de requisitos, seguimiento de *bugs*, o gestión de las pruebas, pero a menudo no están bien integradas. La pobre integración hace que sea más difícil aplicar una metodología afín a través de múltiples proyectos y generar reportes consistentes para conducir la comprensión común del equipo sobre el progreso y la "salud" del proyecto. Esta falta de coherencia puede ser la fuente de procesos de análisis no fiables, lo que hace más difícil de definir y de aplicar mejoras en los procesos en tiempo.

VSTS y TFS proporciona un ambiente integrado que soporta la mayor parte de las actividades del proceso involucrado en el proyecto de desarrollo de *software*. TFS implementa el ciclo de vida de las metodologías a través las plantillas de procesos. Una plantilla de procesos es un conjunto de archivos XML (*Extensible Markup Language*) que proporcionan especificaciones para los procesos y artefactos que conforman una metodología de desarrollo. En este capítulo se abordan cambios significativos realizados a la plantilla de procesos MSF para el desarrollo ágil -v4.0, con el objetivo de personalizarla según las características del proyecto Juegos Consola, la nueva plantilla lleva por nombre: Videojuegos UCI MSF para el desarrollo ágil.

Los procesos definidos en la plantilla de MSF Ágil incorporan ideas claves del movimiento ágil de *software*, junto con los principios y prácticas de MSF. El proceso apoya una estrategia ágil de ingeniería

de *software* que utiliza múltiples iteraciones y un enfoque basado en escenarios para la construcción de aplicaciones.

La plantilla proporciona la automatización y la guía necesaria para apoyar al equipo de desarrollo, incluyendo la gestión de configuración, gestión de proyectos, seguimiento *work ítems* y un portal del proyecto para la comunicación.

3.1 Ajustar el proceso al proyecto

La instrumentación diaria de las actividades y la recopilación automática de datos hace mucho más fácil de seguir un proceso de *software* coherente, VSTS automatiza la guía del proceso y lo instrumenta, de modo que son eliminados los costos asociados con el proceso y la resistencia al cumplimiento en su mayoría.

Cuando se crea un proyecto de equipo, el Asistente para este tipo de proyecto crea cierto número de puntos de atención para centrar los esfuerzos del equipo. Se crea un sitio *web* del proyecto de equipo, que incluye plantillas de documentos e informes predefinidos. También se crea una base de datos de *work ítems* para realizar el seguimiento del trabajo realizado en el proyecto.

VSTS toma en cuenta la diversidad de procesos, permitiéndole al equipo seleccionar o adaptar su metodología a las realidades contextuales. Cuando se comienza un proyecto de equipo en VSTS, se elige la plantilla de procesos.

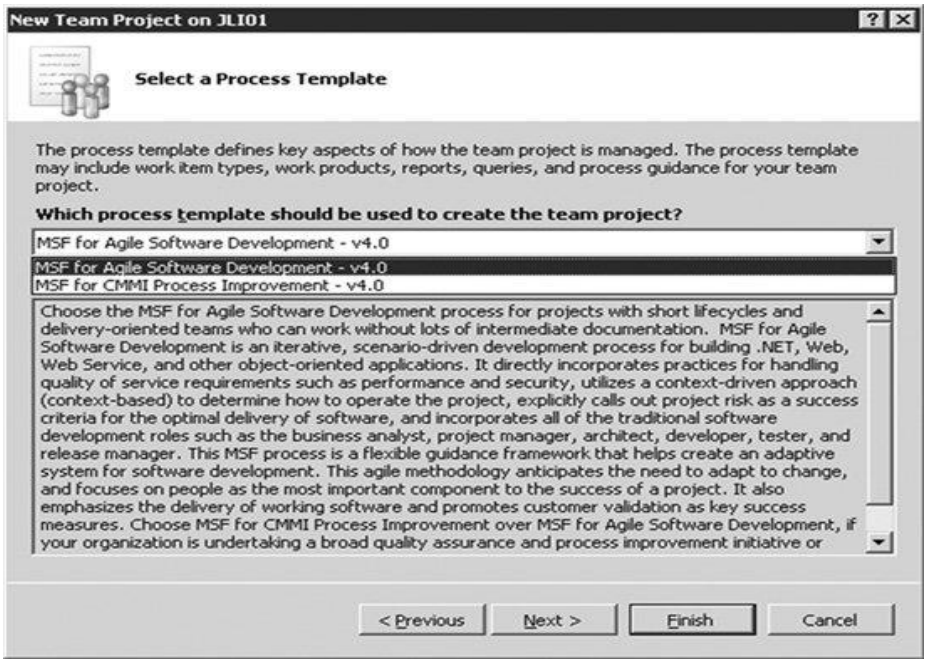

Fig. 3.1 Elección de la plantilla de procesos.

Cuando se crea el Proyecto de Equipo, la primera elección es qué "Plantilla de Proceso" aplicar al proyecto, se seleccionó MSF para el desarrollo ágil -v4.0 predeterminada en el VS 2005 TFS, se propone profundizar en las ventajas del nuevo VS 2008 TFS y la última versión, hasta la fecha, de la plantilla de procesos MSF para el desarrollo ágil v-4.2, con el fin de utilizarlas en futuros proyectos.

Personalización de la plantilla de procesos MSF para el desarrollo ágil

Según (msdn, 2007), las plantillas de procesos definen aspectos claves de un proyecto de equipo que afectan al funcionamiento del mismo. Con la personalización de esta plantilla se puede definir la seguridad para controlar el proyecto de equipo, determinar las plantillas disponibles en el portal de proyecto, las políticas de seguridad para el control de código fuente, nuevos tipos y consultas de *work ítems*, reportes de supervisión y estado, y las iteraciones. La plantilla de procesos define la configuración inicial del proceso para el proyecto de equipo, primero se crea el proyecto de equipo y luego se personalizan las configuraciones.

Para personalizar la plantilla de procesos, primero se descarga del TFS, donde está instalada.

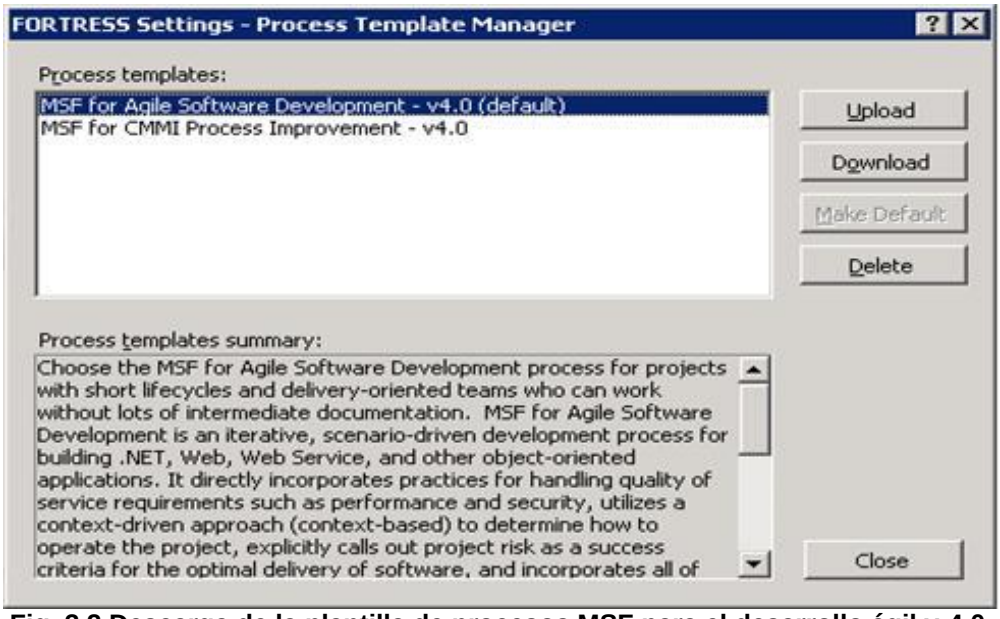

Fig. 3.2 Descarga de la plantilla de procesos MSF para el desarrollo ágil v-4.0.

El proceso de descarga copia todos los archivos de código fuente que define la plantilla de procesos en una carpeta seleccionada.

Fig. 3.3 Plantilla MSF para el desarrollo ágil v-4.0

El tamaño máximo de la plantilla de procesos es 2 GB. Al personalizar la plantilla de procesos MSF para el desarrollo ágil, los cambios no deben exceder ese valor.

El trabajo de personalización tiene más o menos efecto, en dependencia de cómo se realiza. Presenta dos opciones:

- **Personalización para los nuevos proyectos de equipo.**
- **Personalizar un proyecto de equipo existente.**

De las cuales se escoge la primera con el propósito de que la personalización tenga una mayor extensibilidad para el proyecto Juegos Consola, de forma tal que los cambios correspondientes a la

plantilla de procesos de MSF para el desarrollo ágil realizadas en una ubicación, se reflejen en todos los nuevos proyectos de equipo que se realicen a partir de la nueva plantilla de procesos; la segunda opción, permite que los cambios afecten el modo en que el equipo trabaja en ese proyecto pero no debe afectar a otros proyectos existentes o que todavía no se hayan creado, por lo que no es reutilizable.

3.2 Modificaciones necesarias a los complementos de la plantilla de procesos MSF para el desarrollo ágil.

Según el tipo de personalización seleccionada en el epígrafe anterior, su extensibilidad se debe a la personalización de la plantilla de procesos con la modificación de sus **complementos**, los **tipos de *work ítems*** y las **instrucciones de proceso**.

3.2.1 Personalización de complementos.

Los complementos de la plantilla de procesos, son componentes que se ejecutan cuando se crea un nuevo proyecto de equipo. Un complemento instala los ficheros necesarios o configura los datos para su área. *Microsoft* proporciona seis complementos con *Microsoft Visual Studio* 2005 *Team System*: **Seguimiento de *work ítems*, Clasificación, *Windows SharePoint Services*, Control de versiones, Reportes y Grupos y permisos,** como se muestra en la figura 3.4. Todos se pueden modificar o eliminar para la personalización, a excepción del complemento **Clasificación** que sólo puede ser modificado**.**

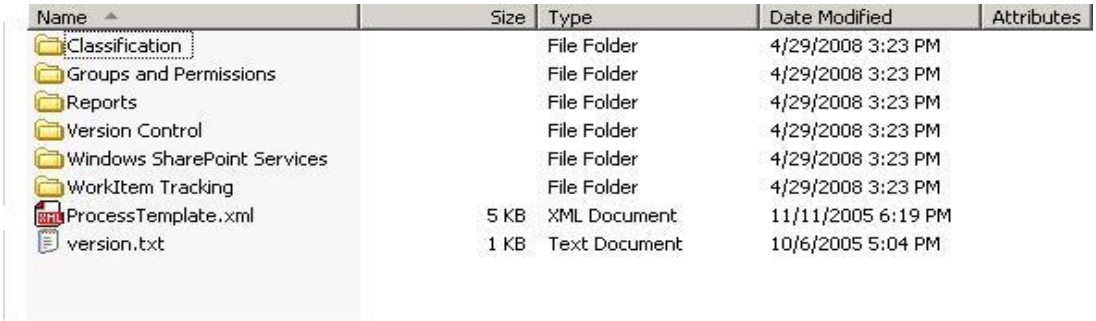

Fig.3.4 Plantilla de Procesos MSF para el desarrollo ágil v-4.0.

Reportes

Los reportes son la principal forma de administración del proyecto y sus líderes se mantienen actualizados respecto al proyecto en curso. Estos reportes están disponibles en el portal *Windows SharePoint Services* o en los nodos de reporte en el *Team Explorer* dentro de *Visual Studio.*

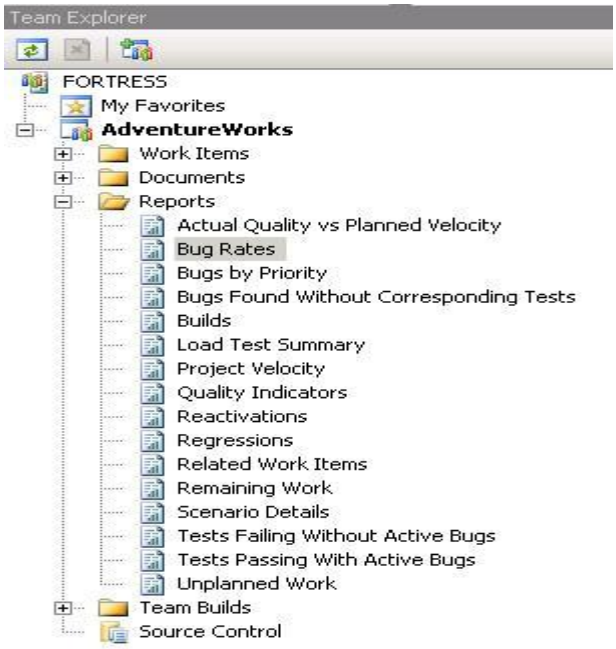

Fig.3.5 Nodos de reporte en el *Team Explorer.*

Cuando se crea un nuevo proyecto de equipo, su Asistente genera un conjunto de informes estándares de acuerdo con las especificaciones de la plantilla de procesos seleccionada. Los reportes se muestran en orden alfabético en *Team Explorer* bajo el nodo Reportes para el proyecto de equipo y los que se ven se abren siempre en modo de sólo lectura. También se pueden agregar los reportes propios que han sido personalizados.

El archivo XML de Informes se denomina ***ReportsTasks*.xml** y se encuentra en la carpeta Reportes de la jerarquía de carpetas de la plantilla de procesos.

Reportes definidos en la plantilla de Procesos MSF para el desarrollo ágil.

En las plantillas de procesos del TFS se definen más de veinte reportes diferentes. De estos reportes solamente nueve pertenecen a la plantilla de procesos MSF para el desarrollo ágil, algunos de los problemas existentes en el proyecto Juegos Consola abordados en el capítulo anterior, evidencian la necesidad de adoptar los reportes de esta plantilla de procesos casi en su totalidad, con el objetivo de observar en tiempo real la "salud" del proyecto para hacer posible la toma de decisiones de manera oportuna, y disminuir el retraso en la entrega del producto final.

Reportes seleccionados para la plantilla de procesos Videojuegos UCI MSF para el desarrollo ágil, necesarios en el proyecto Juegos Consola.

Bugs

Los reportes de *Bug* relacionados en la plantilla de proceso permiten observar qué tipos de *bugs* han sido generados, arreglados y contribuyen a identificar las tendencias del proceso de desarrollo:

***Bugs* por prioridad**. ¿Son encontrados y priorizados los *bugs* correctos? Este reporte muestra la tasa de *bugs* de alta prioridad encontrados contra los *bugs* de baja prioridad.

Los *bugs* reportados por prioridad evalúan la efectividad de la búsqueda y prioridad de los *bugs*. El descubrimiento de *bugs* es normal en el desarrollo del producto. Si en lugar de encontrar *bugs* de alta prioridad, un número desproporcionado de *bugs* de baja prioridad son encontrados, es un indicador de que hay que redireccionar los esfuerzos de las pruebas para buscar los errores de mayor importancia.

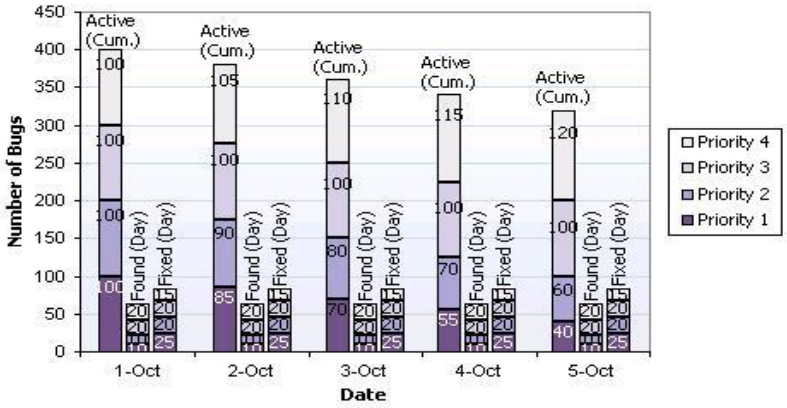

Fig.3.6 Reporte *Bug* por prioridad (MSF *for Agile Software Development Process Guidance*, 2006).

Este gráfico muestra por cada día la acumulación total de *bugs* activos por prioridad y los subtotales diarios encontrados y arreglados.

En la figura las tres series de barras representan los datos siguientes:

- Total de *bugs* activos en el momento del *build*
- Número encontrado en el *build*
- Número resuelto en el *build*

Cada serie se divide en prioridades, de manera que cada barra acumula los *bugs* de mayor a menor prioridad, los de menor prioridad en la parte superior. Si hay muchos *bugs* de alta prioridad activos, hay que asegurarse que exista la capacidad para enfrentarlos. Por otro lado, una significativa escasez de *bugs* de baja prioridad puede conducir también a la insatisfacción del cliente.

Índice de *Bugs*. ¿Con qué efectividad son encontrados los *bugs*, arreglados y cerrados? Este reporte muestra las tendencias en tiempo para los nuevos *bugs*, acumulación de *bugs*, y la resolución de los mismos.

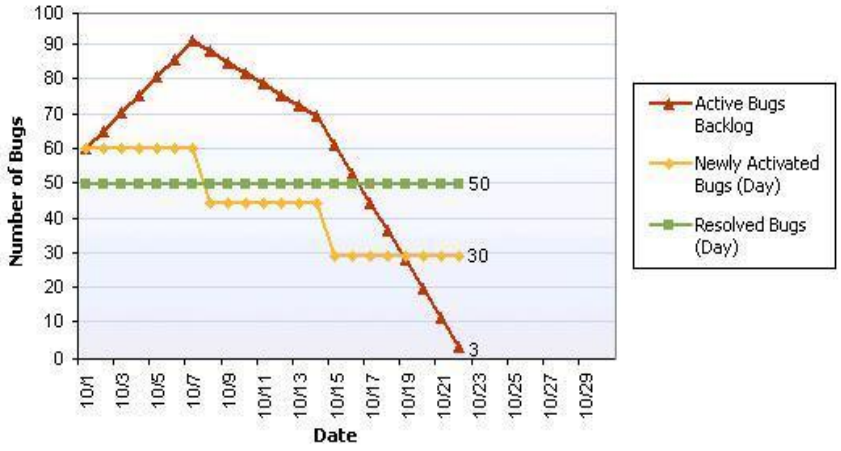

Fig.3.7 Reporte Índices de *Bugs* (MSF *for Agile Software Development Process Guidance*, 2006)

Los *bugs* son una parte fundamental del equipo de trabajo y un área fundamental de riesgo. ¿Cómo determinar con qué efectividad se están encontrando, arreglando y cerrando los bugs? Cuando el índice de arreglados es superior a la tasa de los encontrados, la cantidad de *bugs* activos desciende.

¿Cuál es la cantidad total de *bugs* activos (a veces llamados *bugs* de deuda)?

¿Con qué rapidez son encontrados los nuevos *bugs*?

¿Con qué rapidez son arreglados los *bugs* encontrados anteriormente?

Los índices de *bugs* son mejores interpretados con el conocimiento de todas las actividades en curso del proyecto y de otras métricas de la gráfica que muestra los indicadores de calidad. Un índice alto encontrado puede ser señal de un mal código, código recientemente integrado, pruebas efectivas, o eventos excepcionales. Por otro lado, un bajo índice de *bugs* encontrado puede indicar una solución de alta calidad o ineficacia de las pruebas.

De manera similar un alto índice de *bugs* resueltos es usualmente bueno, pero hay que chequear el progreso del trabajo para asegurar que los *bugs* resueltos han sido prontamente cerrados y comprobar las reactivaciones para asegurar que las soluciones no son prematuras. Se debe usar el código de cobertura, código *churn*, e índices de prueba para ayudar interpretar el significado.

Gestión de *Release*

El informe de gestión de *release* permite juzgar cómo cerrar el *software* que está apto para el *release*. Los siguientes reportes de gestión están disponibles:

Calidad actual contra velocidad planificada. ¿Cuántos escenarios pueden ser completados antes de que la calidad sea inaceptable? Este reporte representa la relación para cada iteración, de dimensión estimada de la calidad, como se muestra a continuación:

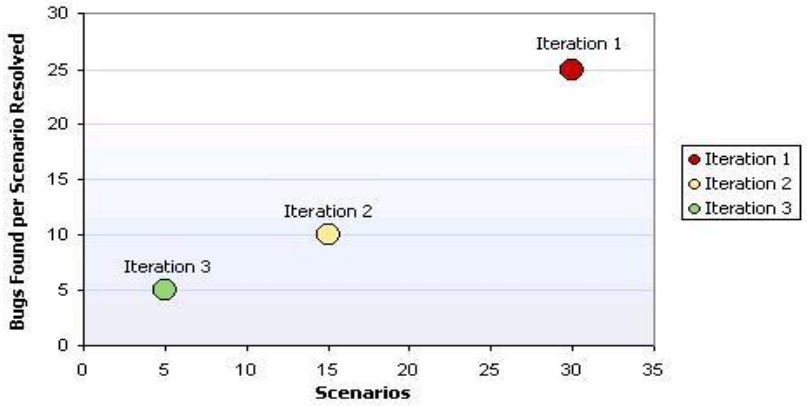

Fig. 3.8 Reporte Calidad actual vs. Velocidad planificada (MSF *for Agile Software Development Process Guidance*, 2006)

Existe una tendencia en el proyecto a “ir rápido”. El objetivo del jefe del proyecto debe ser balancear la calidad y la velocidad planificada para encontrar la proporción máxima de progreso que no afecte la calidad.

El eje ¨**X**¨ es el número de escenarios cerrados realmente (completados) en la iteración. Cada círculo está etiquetado de acuerdo a su iteración.

El eje ¨**Y**¨ es el número total de errores encontrados por escenarios cerrados.

Reporte de Indicadores de calidad. ¿Cuál es la calidad del *software*? Este reporte recoge los resultados de las pruebas, *bugs*, cobertura de código y código *churn* en un único reporte para el seguimiento de la “salud” del proyecto.

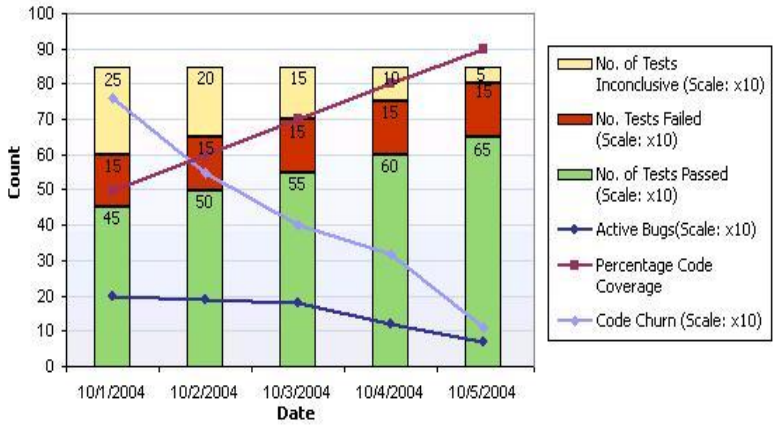

Fig. 3.9 Indicadores de Calidad (MSF *for Agile Software Development Process Guidance*, 2006)

Las barras muestran el número de pruebas que han sido ejecutadas y de éstas, cuántas han resultado pasadas, fallidas o inconclusas.

La primera serie de puntos es la cobertura de código alcanzado en dichas pruebas. Generalmente, a medida que más pruebas son ejecutadas, más código debe ser cubierto.

La segunda serie de puntos es Código *Churn*, es decir, el número de líneas adicionadas y modificadas en el código bajo prueba. Un alto *Churn* indica una gran cantidad de cambios y su efecto secundario, el riesgo de un mayor número de *bugs* introducidos. Por otra parte el alto código *Churn* puede indicar el descenso de la cobertura y la necesidad de volver a escribir las pruebas.

La tercera serie de puntos es la cantidad de *bugs* activos. Debe existir una correlación entre el número de *bugs* activos y el número de pruebas fallidas. Si no desciende la cantidad de *bugs* activos y las pruebas no muestran el fracaso correspondiente, entonces las pruebas probablemente no son de la misma funcionalidad que los *bugs* reportados. Si la cantidad de *bugs* activos disminuyen y los índices de pruebas vencidas no se incrementan, entonces se puede estar en riesgo de un creciente índice de reactivación.

La velocidad es otro de los aspectos claves a medir para la estimación en el proyecto Juegos Consola, el reporte **Velocidad** o “Velocity” permite conocer si el proyecto progresa a la velocidad prevista en aras de terminar en tiempo el producto, y en éste se representa la velocidad con la que el equipo procesa y cierra todos los tipos *work ítems*. Esto muestra la rapidez con que el equipo completa realmente el trabajo

planificado, y cuánto la velocidad varía día a día, o de iteración a iteración. Esta información se usa para planificar la próxima iteración, junto a medidas de calidad. Similar a la gráfica de Tiempo Restante o "*Remaining work*", éste es más útil cuando se observan los días dentro de una iteración o iteraciones dentro de un proyecto.

A diferencia del diagrama de flujo acumulado, el reporte de velocidad muestra la cantidad de *work ítems* resueltos y cerrados cada día.

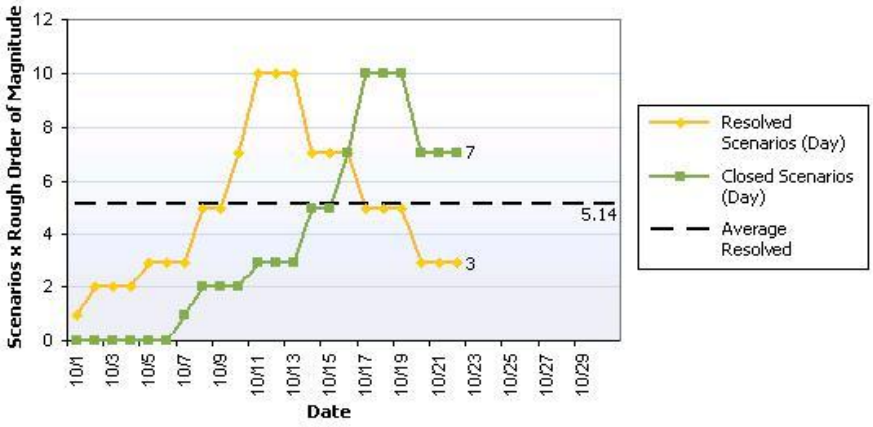

Fig. 3.10 Reporte Velocidad del proyecto (MSF *for Agile Software Development Process Guidance*, 2006)

La acumulación de *bugs* encontrados y los escenarios resueltos es un indicador de calidad que puede advertir qué escenarios han sido resueltos o enviados a prueba antes de tiempo. Este indicador de calidad debe permanecer bajo, si sube se necesita más tiempo para encontrar y darle solución a los *bugs*.

Pruebas

Los reportes de pruebas permiten supervisar la eficacia y el progreso de las pruebas. En la plantilla de procesos MSF para el desarrollo ágil se define el reporte Resumen de la prueba de carga o "***Load Test Summary"***. Este reporte muestra los resultados de las pruebas de carga de la aplicación.

Este reporte puede ser considerado en dos partes. En la parte superior se puede encontrar información tal como: Nombre de la prueba, el usuario de la carga, la duración, tiempo de comienzo, tiempo de terminación, solicitud/segundo, pruebas/segundo, tiempo promedio de pruebas, tiempo promedio de solicitud. También se puede levantar hasta tres cuadros que tienen como máximos 5 solicitudes, pruebas

y operaciones SQL. La segunda parte del reporte es un poco más detallado, brinda mayor información sobre cada una de las pruebas, las solicitudes, las transacciones, los servidores y los errores.

Este informe debe dar una visión instantánea de la prueba de carga y ayudar a facilitar las respuestas a las preguntas de las cuestiones que deben ser analizadas en la aplicación.

Work ítems

Los reportes de *work ítems* permiten evaluar el estado actual del proyecto y el progreso actual del mismo.

Uno de los reportes más importantes para el proyecto Juegos Consola por la posibilidad que brinda de conocer en tiempo real el estado del trabajo, es el de **Trabajo Restante** o "***Remaining Work"***, éste es un diagrama de flujo acumulado que muestra el trabajo restante al medir cómo los escenarios y los *work ítems* de QoS son resueltos y cerrados en la iteración, como se muestra en la figura 3.11. Cada *work ítem* tiene un estado. En MSF para el desarrollo ágil, los escenarios, los cuales son un tipo de *work ítem*, tienen tres estados:

Activos: No han sido codificados.

Resueltos: Han sido codificados, y están listos para ser probados.

Cerrados: Han sido probados y verificados.

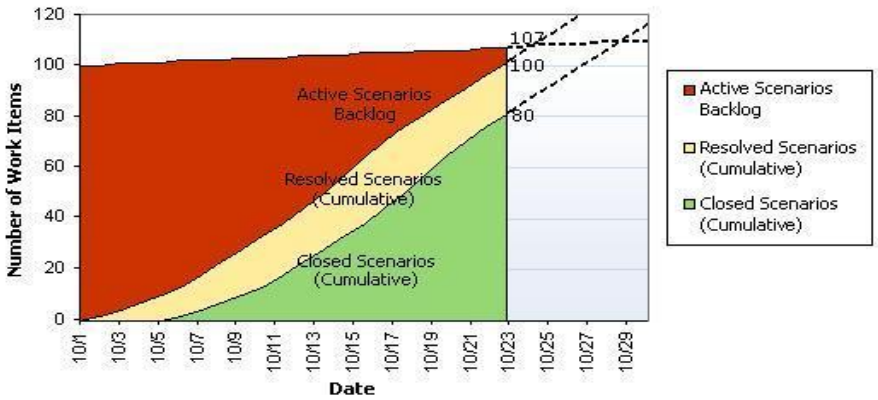

Fig. 3.11 Reporte Trabajo Restante (MSF *for Agile Software Development Process Guidance*, 2006)

Este reporte de Trabajo Restante muestra la tendencia de la "salud" para la entrega final.

Cada serie de datos (banda de color) representa el número de *work ítems* que ha alcanzado el correspondiente estado según la fecha dada. La altura total del gráfico es la cantidad total de trabajo que fue realizado en una iteración. Si la línea superior aumenta, significa que el trabajo total es cada vez mayor, si disminuye, significa que el total de trabajo está disminuyendo, porque el trabajo se está cortando en la iteración.

El estado actual se mide por la altura en una fecha determinada. El resto del retraso se mide por la altura actual de la zona izquierda, es la activa en este caso. El actual complemento se muestra por la altura actual de la zona de la derecha, que es cerrado en este caso. La altura de lo que está entre las bandas indica el trabajo en curso, en este caso los *work ítems*, resueltos pero no cerrados.

Visualmente, es fácil extrapolar una fecha de inventario de terminación o una fecha de terminación de los retrasos acumulados desde un diagrama de flujo acumulado como éste.

Se hace necesario, además, la utilización del reporte **Reactivaciones o "*Reactivations*"** en el proyecto Juegos Consola. Las reactivaciones son *work ítems* que han sido resueltos o cerrados antes de tiempo, un índice alto o un aumento de éste debe advertir que se diagnostique la causa y se le de solución. El índice total de la reactivación de *work ítems*, es mejor cuanto más bajo sea.

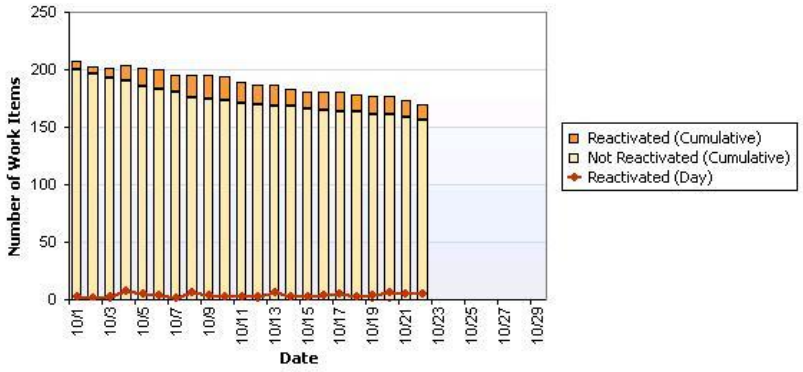

Fig. 3.12 Reporte Reactivaciones (MSF *for Agile Software Development Process Guidance*, 2006)

La altura de la gráfica representa el total de *work ítems* del tipo seleccionado resueltos en el *build* del equipo. La altura del área superior es el número de reactivaciones; es decir, los *work ítems* que ya habían

sido resueltos o cerrados anteriormente, que ahora están activos nuevamente. Y el área inferior es la diferencia entre el total de *work ítems* resueltos y los *work ítems* reactivados.

Se elimina el reporte Trabajo no planeado o "*Unplanned Work*" de la plantilla de procesos Videojuegos UCI MSF para el desarrollo ágil.

En la plantilla de procesos de MSF para el desarrollo ágil se define además, el reporte **"Trabajo no planeado"**, donde su gráfica ilustra la cantidad total de trabajo que se lleva a cabo en la iteración, diferenciando entre el trabajo planeado y el no planeado.

Como en la mayoría de los proyectos, una característica del proyecto Juegos Consola es la imposibilidad de conocer todo el trabajo a realizar antes de cada iteración, incluso dentro de ella. Esto es completamente aceptable si se cuenta con la capacidad de enfrentamiento suficiente para operar con todo el trabajo no planeado, como el tratamiento de *bugs*, pero si no ha sido programada esta capacidad pueden verse obligados a disminuir el trabajo planeado.

El TFS utiliza las iteraciones como una forma de clasificar el trabajo, sin embargo éste no brinda la posibilidad de asociar una fecha específica con una iteración, esto influye negativamente en el reporte "Trabajo no planeado", dado que éste puede ilustrar datos bifurcados por fechas incorrectas de la iteración.

Como parte de la personalización de la plantilla de procesos MSF para el desarrollo ágil, se elimina este reporte del resto definido en la plantilla, según (MSF *for Agile Software Development Process Guidance*, 2006) no se recomienda para proyectos ágiles como es el caso de los videojuegos, además, las deficiencias mencionadas anteriormente pueden desviar el trabajo del equipo, del trabajo planeado, perder la dirección inicial y no concluir en tiempo.

3.2.2 Grupos y permisos.

La seguridad de TFS está basada en los usuarios y grupos. Es posible administrar usuarios y grupos para implementar en el proyecto Juegos Consola un modelo de seguridad que permita a los usuarios tener acceso a los datos y a la funcionalidad que requieran, al mismo tiempo que se protege la información confidencial.

Un grupo es una colección de usuarios que tienen los mismos requisitos de seguridad en TFS. Al agregar usuarios a grupos, se reduce significativamente el tiempo empleado en administrar permisos de usuario. Los usuarios heredan automáticamente los permisos de cualquier grupo al que pertenecen.

Los permisos determinan la autorización para las acciones del usuario, como administración del área de trabajo y creación de proyectos. Cuando se crea un proyecto en TFS, se generan cuatro grupos predeterminados para ese proyecto independientemente de la plantilla de procesos elegida. De forma predeterminada, cada uno de estos grupos tiene un conjunto de permisos definido que rige lo que pueden hacer los integrantes de esos grupos.

El complemento **Grupos y permisos**, define los grupos de seguridad iniciales de un proyecto de equipo y sus permisos. El archivo XML de grupos y permisos se denomina **GroupsandPermissions.xml** y se encuentra en la carpeta *Groups and Permissions* de la jerarquía de carpetas de la plantilla de procesos.

Al crear el proyecto de equipo con la plantilla de procesos de MSF para el desarrollo ágil, se asignan los usuarios a los grupos correspondientes según las funciones de MSF Ágil. Hay dos vías para realizarlo:

- Asignar usuarios a grupos predeterminados basándose en las funciones de MSF Ágil.
- Crear grupos que corresponden a funciones de MSF Ágil y asignar permisos.

Si se asignan usuarios a los grupos predeterminados basándose en sus funciones en MSF Ágil se pierde parte de la granularidad fina del permiso y control que se tiene al crear nuevos grupos basándose específicamente en las funciones. Con el objetivo de alcanzar mayor control sobre las funciones y permisos en el proyecto, se seleccionó la opción de crear nuevos grupos y asociarlos directamente a cada función, asignándole a estos grupos los permisos adecuados en dependencia de las funciones definidas en MSF Ágil. Se agregan permisos para los grupos en los niveles de servidor, proyecto, área y control de código fuente.

Al crear estos grupos personalizados según las funciones de MSF para el desarrollo ágil, hay que tener presente establecer además los permisos adecuados para cada grupo nuevo en *Windows SharePoint Services* o locales en los equipos de TFS*, a*demás de establecer los permisos de TFS.

Según las características del proyecto y los roles definidos por MSF Ágil, se crean los nuevos grupos a los que se le asignan los siguiente permisos en los diferentes niveles, como se muestra en la tabla siguiente.

Roles de MSF Ágil	Servidor	Proyecto	Área	Control de Código Fuente
Arquitecto		Iniciar un *build* Ver información de nivel de proyecto	Editar *work ítems* de este nodo Ver este nodo Ver *work ítems* de este nodo	Lectura Desproteger Proteger Etiquetar Bloquear
Analista de negocios		Ver información de nivel de proyecto	Ver este nodo Ver *work ítems* de este nodo	Ninguno
Desarrollador		Ver información de nivel de proyecto Iniciar un *build*	Editar *work ítems* de este nodo Ver este nodo Ver *work ítems* de este nodo	Lectura Desproteger Proteger Etiquetar Bloquear
Jefe de proyecto		Administrar un build Eliminar este proyecto Editar calidad de	Crear y ordenar nodos secundarios Eliminar este nodo Editar este nodo	Lectura Desproteger Proteger

	build Editar información de nivel de proyecto Publicar resultados de la prueba Iniciar un build Ver información del nivel de proyecto Escribir en el almacén operativo de build	Editar *work ítems* de este nodo Ver este nodo Ver *work ítems* de este nodo	Etiquetar Bloquear Revisar los cambios de otro usuario Desbloquear los cambios de otro usuario Deshacer los cambios de otro usuario Administrar etiquetas Manipular configuración de seguridad Proteger los cambios de otro usuario
Administrador **de *Release***	Administrar un build Eliminar este proyecto Editar calidad de build Editar información de nivel de proyecto Publicar resultados de la prueba Iniciar un build Ver información del nivel de proyecto	Crear y ordenar nodos secundarios Eliminar este nodo Editar este nodo Editar *work ítems* de este nodo Ver este nodo Ver *work ítems* de este nodo	Lectura Desproteger Proteger Etiquetar Bloquear Revisar los cambios de otro usuario Desbloquear los cambios

	Escribir en el almacén operativo de build		de otro usuario Deshacer los cambios de otro usuario Administrar etiquetas Manipular configuración de seguridad Proteger los cambios de otro usuario
Personal de pruebas	Ver información de nivel de proyecto Editar calidad del *build* Publicar resultados de pruebas	Editar *work ítems* de este nodo Ver este nodo Ver *work ítems* de este nodo	Lectura Desproteger Proteger Etiquetar Bloquear

Tabla 3.1 Grupos según los roles de MSF Ágil.

Se propone un grupo adicional para el rol de diseñador por la gran importancia que tiene para el desarrollo de videojuegos, con permisos a nivel de proyecto, tales como: Ver información de nivel de proyecto y publicar los resultados del diseño.

3.2.3 WSS

Según (msdn, 2007) VS 2005 TFS utiliza *Windows SharePoint Services* para posibilitar las funcionalidades del portal del proyecto de equipo. El portal del proyecto es la única ubicación que se puede visitar para obtener información sobre el proyecto de equipo si la plantilla de proceso que se utiliza para crear el proyecto de equipo incluye un portal de proyecto, como en este caso, la plantilla de procesos MSF para el desarrollo ágil.

El complemento ***Windows SharePoint Services*** define el portal del proyecto para el equipo, basado en una plantilla del sitio *SharePoint,* los componentes y el diseño del portal del proyecto están determinados por la plantilla de procesos MSF para el desarrollo ágil. El archivo XML de *Windows SharePoint Services* se denomina ***WssTasks*.xml** y se encuentra en la carpeta *Windows SharePoint Services* de la jerarquía de carpetas de la plantilla de procesos.

El portal del proyecto incluye, noticias sobre el proyecto de equipo, vínculos a los documentos del proyecto de equipo, documentación de la guía del proceso, reporte sobre *work ítems*, *bugs*, *check in* de código, *builds* actualizados, o resultados de pruebas y vínculos a otros ficheros, carpetas o páginas *web*.

El portal del proyecto de equipo es creado como un sitio *web* en *Windows SharePoint Services*, está disponible ya sea desde el *Team Explorer* o desde el navegador, y ofrece a los usuarios la información necesaria para entender rápidamente el estado del proyecto de equipo. Además de la información estándar que aparece en el portal del proyecto, se pueden agregar partes *web* que se conecten a otras fuentes de datos dentro y fuera del proyecto.

Entre los elementos personalizados dentro del WSS, se encuentran: la Librería de documentos, las plantillas de documentos, el directorio Wss de la plantilla y el Fichero *WssTasks.*XML.

Sitios

Un sitio es un grupo de páginas *web* relacionadas. Los sitios *web* basados en *Microsoft Windows SharePoint Services* están diseñados para ser flexibles. Se personaliza el sitio para adaptarlo a las necesidades del equipo. Para personalizar los sitios de *SharePoint*, se deben tener permisos incluidos en los grupos de sitio Diseñador *Web* y Administrador.

El resto de los usuarios del sitio no tienen acceso a las páginas necesarias para realizar estas tareas, salvo que pertenezcan a un grupo de sitio que incluya esos derechos. El sitio puede ser editado con un editor de páginas *web,* compatible con *Windows SharePoint Services*, como *Microsoft Office FrontPage 2003.*

Las Plantillas de *SharePoint* sirven para crear sitios *web* que admiten la funcionalidad de ASP.NET y todas se pueden usar en un servidor en el que se ejecute *Windows SharePoint Services* 3.0, además

permiten crear sitios que ya incluyan contenido de *SharePoint* de colaboración, como listas de *SharePoint* y elementos *web* y bibliotecas de documentos. Los sitios de *SharePoint* ofrecen un elevado nivel de compatibilidad y flexibilidad para interacción y colaboración. Esta auténtica colaboración es posible gracias a *Windows SharePoint Services* 3.0 y ASP.NET 2.0.

El portal definido en la plantilla de procesos se basa en la plantilla de *SharePoint* de Sitio de grupo, para crear un sitio que pueda ser utilizado por el equipo para crear, organizar y compartir información. De esta plantilla se incluye una biblioteca de documentos y las listas básicas.

Plantillas de listas de *Microsoft Office SharePoint Server* 2007

Según (MicrosoftOfficeOnline, 2007) una lista es un conjunto de información que se comparte con los integrantes del equipo. Cuando se crea un sitio de *Microsoft Office SharePoint Server* 2007, se crean automáticamente varios tipos de listas. Se puede personalizar y agregar elementos a estas listas, crear otras a partir de las plantillas disponibles y crear listas personalizadas con la configuración y las columnas que se prefieran.

El tipo de lista que se utilice depende de la clase de información que se desee compartir, se incluyen las listas según sus ventajas tales como: lista de **Anuncios,** para compartir noticias, el estado del proyecto y proporcionar avisos; lista **Calendario,** proporciona una vista visual de los eventos del equipo, además permite un seguimiento de los hitos del equipo; la lista de **Tareas,** para el seguimiento de la información sobre proyectos y para asignar tareas a personas, realizando un seguimiento del estado y el porcentaje de la tarea realizada; lista de **Tareas de proyecto** para almacenar la información de las tareas con un diagrama de *Gantt* con barras de progreso, y hacer el seguimiento del estado del proyecto; y lista **Encuesta,** para recopilar y almacenar comentarios de los miembros del equipo.

Bibliotecas

Una biblioteca es parecida a una lista, excepto que guarda archivos así como información acerca de los mismos. Se puede controlar cómo se ven, administran y crean los archivos en la biblioteca, así como realizar un seguimiento de los mismos. Es posible además, especificar permisos únicos para una biblioteca, o incluso un archivo dentro de una biblioteca.

El tipo de biblioteca que se utiliza depende del tipo de archivo que se comparte, en el caso del proyecto se incluyen las siguientes: la **Biblioteca de documentos,** para archivos varios, incluidos documentos u hojas de cálculo y **Biblioteca de imágenes** para compartir la colección de imágenes digitales o gráficos, por la gran reutilización del equipo de muchos gráficos.

La plantilla MSF para el desarrollo ágil incluye varias bibliotecas de documentos predefinidas para el portal del proyecto y que además pueden ser vistas a través del *Team Explorer* (Ver Fig. 3.13), con los documentos necesarios para el desarrollo del equipo según MSF Ágil. Las bibliotecas de documentos predeterminadas son:

Project Management: Presenta los documentos predeterminados del grupo Administración del Proyecto según MSF Ágil, en la cual se adicionaron las plantillas de los documentos de Gestión de proyecto, del expediente del proyecto Juegos Consola v2.0.

El principal objetivo del grupo de Administración del Proyecto es entregar el producto de valor del negocio dentro de la fecha y el presupuesto acordados. El jefe del proyecto está a cargo de la planificación y la programación de las funciones incluyendo el desarrollo de proyectos y planes de iteración, monitorizando y reportando el estado, e identificando y mitigando los riesgos. También debe realizar consultas con los Analistas del Negocio para planificar los escenarios y QoS para una iteración, consultar con los arquitectos y desarrolladores para la estimación del trabajo, consultar con los probadores para definir el plan de pruebas y facilitar la comunicación con el equipo.

Requerements: Presenta los documentos predeterminados del grupo de Analistas del Negocio según MSF Ágil, en la cual se adicionaron las plantillas de los documentos de requisitos y de aseguramiento de la calidad necesarios, del expediente del proyecto Juegos Consola v2.0.

El principal objetivo del grupo de Analistas del Negocio es definir las oportunidades del negocio. Este grupo trabaja con los *stakeholders* para comprender sus necesidades y objetivos y transformarlos en las definiciones de personas, de escenarios y de requerimientos de calidad de servicio que el equipo de desarrollo usará para construir la aplicación. Además provee de técnicas esenciales al equipo. Este equipo representa la experiencia del usuario y abarca toda la gestión de producto, lo que significa que deben tener en cuenta en todo momento los intereses de los usuarios.

Security: Presenta documentos predeterminados para el grupo de Arquitectos según MSF Ágil. En la misma se adicionaron las plantillas de los documentos de arquitectura y diseño, de despliegue y de análisis de riesgos, necesarios del expediente del proyecto Juegos Consola v2.0.

El principal objetivo del grupo de Arquitectos es garantizar el éxito del proyecto con el diseño de los cimientos de la aplicación. Que incluye tanto la definición de la estructura organizacional de la aplicación como la estructura física del despliegue. En esta misión, el objetivo del arquitecto es reducir la complejidad mediante la división del sistema en particiones claras y simples. La arquitectura resultante es extremadamente importante porque no sólo rige la forma en la que será construido, sino que también determina si la aplicación mostrará las características que son esenciales para el éxito del proyecto. Esto incluye la usabilidad, fiabilidad, sostenibilidad, si cumple con el rendimiento y las normas de seguridad, y si puede ser desarrollado fácilmente frente a los cambios de los requerimientos.

Test: Presenta documentos predeterminados del grupo de Probadores según MSF Ágil, en la cual se adicionaron las plantillas de los documentos de prueba necesarios del expediente del proyecto Juegos Consola v2.0.

El principal objetivo del grupo de Probadores es descubrir y comunicar los problemas con el producto que puedan afectar negativamente su valor. El probador debe comprender el contexto del proyecto y ayudar al resto a tomar decisiones basadas en ese contexto. Un objetivo fundamental para el probador es encontrar y reportar los *bugs* significativos en el producto durante la prueba del mismo. Una vez que un *bug* es encontrado, es también trabajo del probador comunicar con precisión su impacto y describir cualquier vía de solución que pudiera reducir su impacto. El probador realiza las descripciones de los *bugs* y las medias para comprenderlos fácilmente y continuar. Participa con todo el equipo en el establecimiento de las normas de calidad para el producto. El objetivo de las pruebas es demostrar que las funcionalidades están correctas.

El principal objetivo del grupo de Administradores de *Release* es la gestión de la implantación del producto. El administrador de *release* coordina la liberación con operaciones o medios de control. Crean un plan de *release* y certifican el *release* candidato para su envío o despliegue.

Templates: Es una biblioteca para las plantillas de los documentos del equipo

Process Guidance: La guía del proceso para los documentos del equipo.

Además se adicionaron las bibliotecas de documentos:

Implementación: Para las plantillas de los documentos del equipo de desarrolladores, relacionadas con la implementación y el soporte del proceso de desarrollo, necesarios según el expediente del proyecto Juegos Consola v2.0.

El objetivo fundamental del grupo de Desarrolladores es implementar la aplicación, tal como se especifica en los plazos previstos. El desarrollador también debe ayudar a especificar las características del diseño físico, estimar tiempo y esfuerzo para completar cada iteración, construir o supervisar la ejecución de funciones, preparar el producto para su despliegue, y proporcionar la tecnología.

Documentos legales: Para las plantillas de los documentos legales del equipo, necesarios según el expediente del proyecto Juegos Consola V2.0.

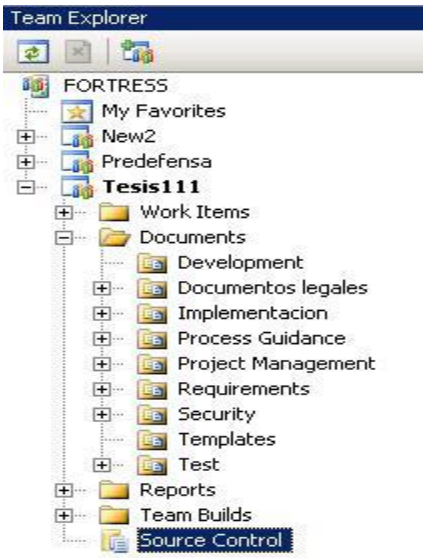

Fig.3.13 Librerías de documentos de la plantilla de proceso Videojuegos UCI MSF para el desarrollo ágil.

3.2.4 Complemento Clasificación

El complemento **Clasificación** controla las iteraciones y la estructura de un proyecto de equipo. El archivo XML de Clasificación se denomina ***Classification*.xml** y se encuentra en la carpeta Clasificación de la jerarquía de carpetas de la plantilla de procesos.

Las iteraciones que define un proceso determinan cuántas veces tendrá que repetir el equipo un conjunto determinado de actividades principales (como planear, desarrollar, probar). Afectan a las consultas e informes del *work ítem* porque se utilizan para agrupar *work ítem*.

Las áreas representan grupos clave en el proyecto de equipo. Se utilizan para agrupar *work ítems* para consultas e informes específicos. Las áreas aparecen en el campo *System.AreaPath* de cualquier *work ítem* que haga referencia a este campo.

El número de iteraciones es específico para el proceso que se desarrolla, ya que se encuentra en correspondencia directa con la complejidad del proceso. Se considera que no es factible definir un número de iteraciones para un proyecto futuro del cual no se conoce la complejidad, por ello se recomienda que se personalice este complemento según el proceso que se esté llevando a cabo y los *work ítems* que se necesiten agrupar para las consultas e informes requeridos.

3.2.5 Seguimiento de *work ítems*.

El complemento **Seguimiento de *work ítems*** define los tipos de *work ítems* iniciales de un proyecto de equipo, las consultas y las instancias del *work ítems*. El archivo XML de Seguimiento de *work ítems* se denomina ***workitems.xml*** y se encuentra en la carpeta Seguimiento de *work ítems* de la jerarquía de carpetas de la plantilla de procesos.

En el archivo XML, se especifica una o más tareas y sus dependencias. Se pueden especificar tres tipos clave de tareas: tipos de *work ítems*, consultas de *work ítems* e instancias de *work ítems*.

Tipos de *work ítems*

Los miembros del equipo utilizan los *work ítems* para realizar el seguimiento de trabajo del trabajo del proyecto. Un tipo de *work ítems* es una plantilla a partir de la que se crean nuevos *work ítems* de ese tipo,

por lo que al crear un *work ítem*, éste contiene los campos y el comportamiento definidos en el tipo de *work ítems* desde el que ha sido creado. Es posible expandir la selección de *work ítems* de VSTS de modo que se mejore la comunicación y la atención a los procesos que utiliza el equipo.

Un tipo de *work ítems* define las reglas, campos, estados y transiciones de un *work ítem* del que se efectuará un seguimiento en un proyecto de equipo, como *bugs*, requisitos y riesgos. El tipo de *work ítems* se especifica en un archivo XML de definición de tipos, en la carpeta Seguimiento de *work ítems*, de la carpeta *TypeDefinitions*. En la plantilla de procesos MSF para el desarrollo ágil se definen los tipos de *work ítems* en los archivos *Bug.xml*, *QoS.xml*, *Scenario.xml*, *Risk.xml* y *Task.xml*, respectivamente. Y es posible personalizar o crear nuevos archivos de definición de tipos en la carpeta *TypeDefinitions*.

Los cinco tipos de *work ítems* definidos en la plantilla de procesos facilitan al equipo de desarrollo de videojuegos administrar eficientemente su trabajo, por lo que en el proceso de personalización que se lleva a cabo no se requiere la modificación de los tipos de *work ítems*. Pero se podría expandir la selección de *work ítems* en futuros procesos de desarrollo de videojuegos si lo cree necesario el Jefe de proyecto, de modo que atienda mejor los procesos que utiliza el equipo y el modo de comunicarse, para lo que se podrían crear nuevo tipos de *work ítems* o modificar los existentes. Se propone crear nuevos tipos de *work ítems* para un *bugs* específico del proyecto en el momento en que se esté desarrollando alguna aplicación, una solicitud de cambio, un requisito de calidad de servicio, un riesgo que haya que controlar o una tarea basada en una situación concreta. De igual forma se pueden agregar campos a un tipo de *work ítems* o cambiar el comportamiento del flujo de trabajo de los mismos según las necesidades más específicas.

Consultas de *work ítems*

Es posible ejecutar una consulta de *work ítems* para encontrar agrupaciones concretas de *work ítems*, como riesgos o *bugs* activos. Las consultas de *work ítems* se especifican en archivos de consulta de *work ítems* (WIQ) en la subcarpeta *Queries* de la carpeta *Work Ítem Tracking*, situada en la carpeta donde se realizó la descarga de la plantilla de procesos.

En la personalización de la plantilla de proceso Videojuegos UCI MSF para el desarrollo ágil no se requiere la modificación de las consultas de *work ítems*, ya que las consultas definidas en la plantilla

cubren la búsqueda de los *work ítems* específicos necesarios para el proceso de desarrollo de videojuegos que se lleva a cabo actualmente. Se propone para futuros proyectos valorar la opción de crear nuevas consultas según los problemas y las necesidades que se presenten.

Entre las consultas de *work ítems* que MSF Ágil provee para apoyar las actividades y funciones del proceso:

Los *Bugs* activos, todos los *work ítems* del proyecto de equipo, todos los escenarios, todos los *work ítems*, mis *work ítems*, la lista de chequeo del proyecto, los *bugs* resueltos, todas las tareas, todos los QoS, los *bugs* no priorizados y todas las cuestiones.

3.2.6 Control de versiones del *Team Foundation Server*

El complemento de **Control de versiones** define los permisos de seguridad para el control de versiones iniciales de un proyecto de equipo, las notas de protección y si es necesaria la desprotección exclusiva. El archivo XML del control de versiones se denomina ***VersionControl*.xml** y está ubicado en la carpeta Control de versiones de la jerarquía de carpetas de la plantilla de procesos.

Notas de protección

El desarrollador proporciona las notas de protección cuando protege el código para describir de qué modo los cambios de código afectan a los procesos del equipo, en caso de que lo hagan.

Se utiliza el elemento *checkin_note* para definir una nota de protección.

<checkin_note label="" required="" order=""/>

Atributos del elemento *checkin_note*:

Label: Etiqueta que describe la nota de protección. La etiqueta se muestra en el cuadro de diálogo Protecciones pendientes cuando los miembros del equipo realizan una protección.

Required: Especifica si la nota de protección debe tener un valor. Si se establece como verdadero, la nota de protección debe tener un valor. Si se establece como falso, el valor es opcional.

Order: Especifica un número ordinal que indica el orden en que se van a mostrar las notas de protección, este atributo es opcional.

A modo de ejemplo se muestra cómo personalizar las notas de protección de MSF para el desarrollo ágil para proporcionar una nota de protección adicional denominada "*Documentation Impact*", que no requiere un valor.

De esta forma quedó el archivo *VersionControl*.xml de la plantilla de procesos Videojuegos UCI MSF para el desarrollo ágil, por el impacto que pudieran ocasionar las modificaciones del código en la documentación del mismo, dada la necesidad de realizar esta actividad durante el desarrollo de videojuegos.

```
<taskXml>
  <checkin_note label="Code Reviewer" required="false" order="1"/>
  <checkin_note label="Security Reviewer" required="false" order="2"/>
  <checkin_note label="Performance Reviewer" required="false" order="3"/>
  <checkin_note label="Documentation Impact" required="false"/>
</taskXml>
```

Para futuros proyectos se propone que se personalicen las notas de protección según las dificultades y necesidades que se presenten en el trascurso del mismo.

Desprotección exclusiva

Es posible controlar si varios usuarios pueden desproteger un archivo al mismo tiempo. Se utiliza el elemento *exclusive_checkout* para especificar las propiedades de desprotección.

```
<exclusive_checkout required=""/>
```

Si el atributo *required* se establece como verdadero, sólo una persona podrá desproteger un archivo en cada ocasión. Si se establece como falso, varias personas podrán desproteger al mismo tiempo un archivo y se deberán resolver las modificaciones realizadas cuando se proteja el archivo.

En la plantilla de procesos Videojuegos UCI MSF para el desarrollo ágil se especifica que es necesaria la desprotección exclusiva, de la forma:

<exclusive_checkout required="true"/>

De manera que sólo un miembro del equipo pueda desproteger un archivo en cada ocasión, y se propone que se entrene al equipo de desarrollo del proyecto para trabajar de forma tal, que varias personas puedan desproteger al mismo tiempo el archivo, y alcanzar así mayor agilidad.

Permisos

El control de versiones cuenta con un conjunto concreto de permisos que se puede configurar para una plantilla de procesos. Al especificar los permisos se definen las acciones que pueden realizar los individuos y grupos de seguridad en los elementos sujetos al control de versiones.

Se utiliza el elemento *permission* para conceder, negar, o conceder y negar permisos para una identidad, de la forma:

<permission allow="" identity=""/>
<permission deny="" identity=""/>
<permission allow="" deny="" identity=""/>

Permisos:

Read: Permite leer el contenido de un archivo o carpeta.

PendChange: Permite desproteger, agregar, eliminar, bifurcar, combinar, recuperar y realizar otras actividades asociadas a un conjunto de cambios.

Checkin: Permite proteger los cambios.

Label: Permite aplicar una etiqueta a los elementos.

Lock: Permite bloquear un elemento para que otros no puedan actualizarlo.

ReviseOther: Permite cambiar el contenido de los comentarios del conjunto de cambios y las notas de protección de otro usuario.

UnlockOther: Permite quitar el bloqueo de otro usuario.

UndoOther: Permite deshacer los cambios pendientes de otro usuario.

LabelOther: Permite modificar la etiqueta de otro usuario.

AdminProjectRights: Permite establecer la configuración de seguridad para el control de versiones.

CheckinOther: Permite realizar la protección como otro usuario. Este permiso es necesario con utilidades de conversión. En esta plantilla se asignaron los permisos correspondientes a los nuevos grupos creados según la Tabla 3.1.

En el ejemplo siguiente se muestra cómo conceder permisos para que el grupo ***Arquitecto*** pueda modificar archivos sujetos al control de versiones.

```
<taskXml>
  <permission allow="Read, PendChange, Checkin, Label, Lock"
   identity="[$$PROJECTNAME$$]\Arquitecto"/>
</taskXml>
```

3.3 Instrucciones sobre el proceso

Las instrucciones del proceso explican las funciones, *work ítems*, productos de trabajo (documentos y plantillas), actividades y reportes para un proceso de desarrollo de *software* concreto, y constituyen una colección de páginas *Web* HTML almacenadas en el portal del proyecto de equipo. De esta forma complementan la plantilla de procesos.

En el VSTS la orientación del proceso está completamente integrada a las herramientas de desarrollo. A la guía de proceso se puede acceder desde el portal del proyecto, desde *Team Explorer*, desde las listas de elementos de trabajo de *Microsoft Excel* y desde los planes de *Microsoft Project*.

Al adaptarse la plantilla de procesos MSF para el desarrollo ágil al proyecto Juegos Consola, el equipo requiere de la personalización de las instrucciones de proceso. A medida que se modifican los complementos de la plantilla, el contenido HTML de la guía de procesos se actualiza, de esta forma el equipo cuenta con la orientación correcta para el desarrollo de videojuegos.

En la orientación del proceso se explican todos los componentes para ayudar al equipo a coordinar los esfuerzos y seguir su proceso. En la siguiente se expone la información de la orientación del proceso que se proporciona para cada elemento de proyecto de equipo.

Elemento de proyecto de equipo	Información de la orientación del proceso
Informes	Explica las distintas series de datos disponibles en los informes, muestra ejemplos de informes con errores y sin ellos.
Elementos de trabajo	Explica cada uno de los *work ítems,* sus campos*,* los estados disponibles y su significado, las transiciones de estado válidas y los motivos para usarlas.
Consultas	Breve explicación de las consultas de cada *work ítem* que se pueden realizar en el proyecto de equipo.
Productos de trabajo (documentos y plantillas)	Explica la finalidad de cada uno de los productos de trabajo y proporciona un vínculo a la plantilla de productos de trabajo.

Tabla 3.2 Información de la orientación del proceso para cada elemento de proyecto de equipo.

En las instrucciones del proceso además, se describe el proceso del equipo utilizando un marco de trabajo conceptual coherente. El marco de trabajo se compone de *work ítems*, cada uno de los cuales representa una parte distinta del proceso. Cada elemento de contenido tiene su propia página en la que éste puede verse. Entre los elementos de contenido de la orientación del proceso se cuenta con: **Información general,** que introduce conceptos fundamentales como ciclos e iteraciones, los principios y los patrones de comportamiento; **Función,** describe las responsabilidades que asumirán uno o varios miembros del equipo en el proyecto; **Secuencia de trabajo,** que es una colección de actividades relacionadas que

tienen una finalidad o función común; **Actividad,** cada actividad tiene criterios de entrada y criterios de salida y el elemento **Cómo,** que describe un procedimiento para realizar una tarea en las herramientas.

Según las modificaciones realizadas a la plantilla de proceso, se elimina de la guía la información publicada referente al reporte **"Trabajo no planeado"**, al ser éste eliminado del conjunto de reportes favorables para el progreso y buen funcionamiento del proyecto. Y se añade el reporte **"Resumen de la prueba de carga".**

Además se publica el resto de los documentos necesarios para la documentación del proceso, según el expediente del proyecto Juegos Consola V2.0, para cada uno de los roles específicos de MSF Ágil.

Consideraciones del Capítulo

Como resultado de esta investigación se obtiene la primera versión de la plantilla **Videojuegos UCI MSF para el desarrollo ágil** a partir de la personalización de la plantilla de procesos MSF para el desarrollo ágil, por la vía seleccionada de **Personalización para los nuevos proyectos de equipo.** Luego de realizar las modificaciones requeridas a los seis complementos, los tipos de *work ítems* y las instrucciones de proceso, se espera que esta nueva plantilla tenga una mayor extensibilidad para el proyecto Juegos Consola, pues los cambios correspondientes a la plantilla de procesos de MSF para el desarrollo ágil se reflejarán en todos los nuevos proyectos de equipo que se creen a partir de la nueva plantilla de procesos. Con la aplicación de la misma se prevé que disminuyan los problemas de calidad y tiempo de entrega, existentes actualmente en la realización de videojuegos en el proyecto Juegos Consola.

CONCLUSIONES

Posteriormente al estudio del contenido expuesto en (Bauta, et al., 2007), de las fuentes bibliográficas consultadas, de la investigación realizada en el primer capítulo y de la comparación realizada en el segundo, se reafirma lo expresado en (Bauta, et al., 2007) de que MSF Ágil es el marco de trabajo que más se ajusta a las características del proyecto Juegos Consola de la Facultad 5 de la UCI, para su aplicación se llevó a cabo como solución técnica la personalización de la plantilla de procesos MSF para el desarrollo ágil, y se concluye que éste es el modelo más adecuado para mejorar la situación existente actualmente en proyecto Juegos Consola. Debido fundamentalmente a la gran flexibilidad y adaptabilidad que presenta, además de arrojar resultados en tiempo récord y ofrecer comodidad a los miembros del quipo.

El VSTS es una plataforma para un conjunto de herramientas integradas y extensibles, ésta permite agilizar el trabajo con la implementación del proceso de desarrollo a partir de la plantilla MSF para el desarrollo ágil, que viene predeterminada en el VSTFS. VSTS incorpora funcionalidades básicas e intuitivas a todas las áreas de gestión del ciclo de vida de las aplicaciones para ayudar a resolver problemáticas de negocio y retos de desarrollo, funcionalidades que abarcan desde el modelado inicial y el diseño, la gestión de cambios, las pruebas y la implantación. Incorpora conectividad, ya que ha sido construido desde cero para trabajar con él como un sistema integrado. VS 2005 TFS desde su implantación en el proyecto ha favorecido la centralización del almacenamiento de los datos relativos al proyecto entre ellos las métricas del proyecto para generar los reportes sobre el estado del proyecto en tiempo.

Con la personalización de la plantilla de procesos MSF para el desarrollo ágil que tiene como resultado la plantilla de procesos Videojuegos UCI MSF para el desarrollo ágil, debe mejorar notablemente la coordinación, la gestión, la colaboración del equipo, la toma oportuna de decisiones, y por ende la calidad del producto y la entrega del producto final en tiempo. Todos los nuevos proyectos que se creen para el desarrollo de videojuegos en el proyecto Juegos Consola pueden basarse en esta nueva plantilla creada: Videojuegos UCI MSF para el desarrollo ágil, y personalizarse aún más según el proceso que se lleve a cabo en ese momento.

RECOMENDACIONES

1. Fomentar la investigación en aquellos proyectos que aún se rigen por las metodologías tradicionales de desarrollo de *software*, para determinar si la metodología que se está usando es la que realmente se ajusta a las características del proyecto, del equipo y del *software* que se produce.
2. Fomentar la investigación de herramientas integradas basadas en la denominación *software* libre para utilizarlas en los proyectos de la UCI, y que sirvan de alternativa al VSTS y al TFS. Así como el uso de herramientas para una mejor edición de las plantillas de proceso como es el caso de *Process Template Editor.*
3. Extender el uso de entornos integrados de desarrollo en todos los proyectos de la UCI por las grandes ventajas y facilidades que brindan para la colaboración y el buen desenvolvimiento del equipo.
4. Desarrollar en la UCI proyectos para la realización de herramientas de gestión integradas al proceso de desarrollo basadas en la denominación *software* libre y que permitan la aplicación de metodologías ágiles.
5. Personalizar aún más la plantilla Videojuegos UCI MSF para el desarrollo ágil según las necesidades que se presenten en el proyecto Juegos Consola durante el desarrollo de un nuevo proyecto.

REFERENCIAS BIBLIOGRÁFICAS

Bauta, Reina Mercedes and Castillo, Dallandy. 2007. *Propuesta de estrategia ágil para el desarrollo de video juegos.* Ciudad de la Habana : UCI, 2007.

Brooks, Frederick. 1987. No Silver Bullet. *The Mythical Man-Month:Essays on Software Engineering.* 1987.

Colombia, M c. 2000. *Microsoft Solution Framework(MSF):Disciplinas y buenas prácticas para el desarrollo e implantación de proyectos.* 2000.

Consultores, G. 2006. Disciplina de administración del proyecto - M.S.F. [Online] 2006. Disponible en: http://www.gpicr.com/msf.aspx.

Fowler, M. 2003. *La Nueva Metodología.* 2003.

Fraile, Luis and De La Torre, Cesar. 2005. *Metodologías Agiles, MSF-Agile y Team System 2005.* [Online] 2005.

Disponible en: http://209.85.165.104/search?q=cache:tZl6zvYpcL4J:download.microsoft.com/download/7/9/9/79981b45-cdb0-4958-9c0a-6db02ddbb976/Metodologias_Agile_MSF_Agile_y_TEAM_SYSTEM_2005.ppt+ Metodologias+Agiles,+MSF-Agile+y+team+system+2005&hl=es&ct=clnk&cd=1&gl=cu.

Glosario.Net. 2006. Diccionario:Términos técnicos de Internet. [Online] 2006. Disponible en: http://tecnologia.glosario.net/terminos-tecnicos-internet/videojuegos-1721.html.

Guckenheimer, Sam. 2006. *Software Engineering with Microsoft Visual Studio Team System.* 2006.

GUIDANCE, P. 2005. [Online] 2005.

Disponible en: http://www.microsoft.com/downloads/details.aspx?familyid=9F3EA426-C2B2-4264-BA0F-35A021D85234&displaylang=en.

IEEE. 1993. *IEEE Software Engineering Standar :Glossary of Software Engineering Terminology.* s.l. : IEEE Computer Society Press, 1993.

Imaginet Resource Corp. *ImagiNET.* [Online] Disponible en: http://www.imaginets.com/Default.aspx?tabid=133.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 1999. *Proceso Unificado de Desarrollo de Software.* 1999.

Letelier, Patricio and Penadés, M.C. 2006. *Metodologías ágiles para el desarrollo de software:Extreme Programming(XP).* 2006.

Letelier, Patricio. 2006. Proceso de desarrollo de software. [Online] 11 2006. Disponible en: https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20Proceso%20de%20Desarrollo%20de%20SW.doc

Maddison. 1983. INGENIERIA DE SOFWARE EDUCATIVO. [Online] 1983. Disponible en: http://www.fi.uba.ar/laboratorios/lsi/c-icie99-ingenieriasoftwareeducativo.pdf.

Medela, S P. 2006. ¿Y con UML que hacemos? [Online] 12 2006. Disponible en: http://www.mug.org.ar/Descargas/Jornadas/2481.aspx.

MicrosoftOfficeOnline. 2007. Microsoft Office Online. *Microsoft Office Online.* [Online] 2007. Disponible en: http://office.microsoft.com/es-es/getstarted/HA100738473082.aspx.

2007. msdn. [Online] 2007. Disponible en: http://msdn2.microsoft.com/es-es/library/ms194945(VS.80).aspx.

2006. MSF for Agile Software Development Process Guidance. *Microsoft.* [Online] 2006. Disponible en: http://www.microsoft.com/downloads/details.aspx?familyid=9F3EA426-C2B2-4264-BA0F-35A021D85234&displaylang=en.

2005. *Navegapolis.com.* [Online] abril 23, 2005. Disponible en: http://www.navegapolis.net/content/view/95/87/.

Peppard, Joe and Rowland, Phillip. 1996. La esencia de la reingeniería en los procesos de negocios. [Online] 1996.

Pressman, Roger S. 1997. *Ingeniería del Software: Un enfoque práctico.* s.l. : McGraw Hill, 1997.

Rizzo, Thomas. 2007. Microsoft Tech net. *Información general sobre Microsoft Office SharePoint Server 2007* . [Online] 2007. Disponible en: http://technet.microsoft.com/es-es/magazine/cc162511.aspx.

Sánchez, J P. 2004. Metodologías Agiles: La ventaja competitiva de estar preparado para tomar decisiones lo más tarde posible y cambiarlas en cualquier momento. [Online] 2004. Disponible en: http://www.agile-spain.com.

Sommerville, Ian. 2002. Ingeniería de Software. Mexico : Pearson Education, 2002.

Spaces, W L. 2005. Que hay para decir de MSF(Microsoft Solution Framework). [Online] 2005.

BIBLIOGRAFÍA

CORPORATION, M. 2007. Disciplina para la administración de proyectos MSF v. 1.1, 2007. Disponible en: http://www.microsoft.com/latam/technet/articulos/200304/art01/

CORPORATION, M. 2007. *Process Templates,* 2007. Disponible en: http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx.

Desarrollo de Software. 2006, [PDF]. 2004. Disponible en: http://www.fi.uba.ar/materias/7500/schenone-tesisdegradoingenieriainformatica.pdf

Fernández, L. 2005. Herramientas y métricas para guiar inspecciones y pruebas de software. [Online] 2005. Disponible en: http://web.iti.upv.es/~squac/JTS/JTS2005/contenido.html.

Jfrrer .2005. Principios Ágiles. Disponible en: http://www.agile-spain.com/agilev2/principios_agiles

Lozada's, J. C. 2004. La nueva estrategia de Control de Versiones de *Microsoft: SourceSafe* y parte de *Visual 2005 Team System,* 2004. Disponible en: http://blogs.msdn.com/juanlozv/

Martin, J. 1991. *Rapid Application Development.* New York : Macmillan Inc., 1991.

Reynoso, C. 2004. *Métodos Heterodoxos en Desarrollo de Software* . 2004.

S.A, I. P. G. 2007. Presentación de Metodología MSF *(Microsoft Solutions Framework),* 2007]. Disponible en: http://www.e-gattaca.com/eContent/library/documents/DocNewsNo50DocumentNo6.PDF

2007. Team Fundation Server Administration Tools. *CodePlex.* [Online] 2007. Disponible en:http://www.codeplex.com/Wiki/View.aspx?ProjectName=TFSAdmin.

Visual Studio 2005 Team Foundation Server Power Tools. *MSDN*. [Online] Disponible en: http://msdn2.microsoft.com/en-us/teamsystem/aa718351.aspx.

---. *Visual Studio Enterprise Tools Visual Studio 2005 Team System: Microsoft Solutions Framework.* 2007 [2007]. Disponible en: http://msdn2.microsoft.com/en-us/library/aa302179.aspx

---. *Visual Studio Team System*, 2007 [2007]. Disponible en: http://www.microsoft.com/events/series/Msdnvsts.aspx#TeamFoundationServer

---. *Visual Studio Team System*

---. *Visual Studio Team System Microsoft Solutions Framework Team Model y Team System* 2007. [Disponible en: http://msdn2.microsoft.com/es-es/library/ms195024(VS.80).aspx

GLOSARIO

1. **Adaptabilidad.** Facilidad con la que un sistema o un componente puede modificarse para corregir errores, mejorar su rendimiento u otros atributos, o adaptarse a cambios del entorno. Ver también: escalabilidad.
2. ***Blog*.** En español bitácora, es un sitio *web* periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente. El término proviene de las palabras *web* y *log* ('log' en inglés = diario).
3. **Ciclo de vida.** Período de tiempo que comienza con la concepción del producto de software y termina cuando el producto esta disponible para su uso.
4. **CMMI.** Siglas de "*Capability Maturity Model Integration*", modelos desarrollados por SEI que Integran varias disciplinas: Desarrollo de *software*, Ingeniería de sistemas, Integración de productos y procesos de desarrollo.
5. **Escalabilidad.** Facilidad con la que un sistema o un componente puede modificarse para aumentar su capacidad funcional o de almacenamiento. Ver también: adaptabilidad.
6. **Extreme Programming.** Metodología heterodoxa de programación. Es la más popular de las denominadas metodologías ágiles. Surgida a partir de la metodología de trabajo empleada *Kent Beck, Wark Cunningham* y *Martin Fowler* en el desarrollo del proyecto C3 para *Chrysler*. *Extreme Programming* (XP) se funda en cuatro valores: comunicación, simplicidad, *feedback* y coraje.
7. **Flexibilidad.** Facilidad con la que un sistema o un componente puede modificarse para ser empleado con aplicaciones o en entornos distintos para los que fue construido.
8. **GUI.** Siglas de **"***Graphical User Interface".* Es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.
9. **IEEE.** Siglas de "*The Institute of Electrical and Electronics Engineers".*
10. **Ingeniería del *software*.** (1) Aplicación de procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de software. (2) El estudio de la aplicación (1).
11. **Interoperabilidad.** se define como la habilidad que tiene un sistema o producto para trabajar con otros sistemas o productos sin un esfuerzo especial por parte del cliente. Es la capacidad de los

sistemas de tecnologías de la información y las comunicaciones (TIC), y de los procesos empresariales a los que apoyan, de intercambiar datos y posibilitar la puesta en común de información y conocimientos.

12. **J2EE .***Java Platform, Enterprise Edition* o Java EE (anteriormente conocido como Java 2 *Platform, Enterprise Edition* o J2EE hasta la versión 1.4), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.
13. ***Mainsoft Corporation***. empresa especializada en la adaptación de productos de *software*, permite a sus clientes desarrollar aplicaciones de *software* y portarlas posteriormente a entornos Unix fácilmente. Tiene su sede en San José (California) y cuenta con delegaciones en Chicago, *Washington* D.C., Londres e Israel.
14. **Metodologías ágiles.** Estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas; centradas en las personas o los equipos, iterativas, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa de cliente.
15. ***Plugin.*** Componente enchufable (o *plug-in* -en inglés "enchufar") es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica. Ésta aplicación adicional es ejecutada por la aplicación principal.
16. ***Release:*** Lanzamiento de la versión definitiva de un producto final a menos que aparezcan errores que lo impidan.
17. **SOAP.** (*Simple Object Access Protocol*) es un protocolo estándar creado por *Microsoft*, IBM y otros, define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios *Web*.
18. ***Stakeholders.*** Partes involucradas en el proceso de desarrollo de un *software* que esperan recibir un beneficio del mismo. (usuarios, clientes, *managers*).
19. ***Wiki*.** Una *wiki*, es un sitio *web* cuyas páginas *web* pueden ser editadas por múltiples lectores a través del navegador *web*. Los usuarios pueden crear, modificar o borrar un mismo texto que comparten. Los textos o "páginas *wiki*" tienen títulos únicos. Si se escribe el título de una "página-wiki" en algún lugar del *wiki*, esta palabra se convierte en un "enlace *web*" a la página *web*.

20. **XML**. Sigla en inglés de *Extensible Markup Language* («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathM.

1