UNIVERSIDAD NACIONAL DE LUJÁN

Reducción de la volatilidad de requerimientos en proyectos de desarrollo de Software

Febrero 2014

LOISA, Hernán Esteban
Legajo 35459

Director
PANESSI, Walter

Licenciado en Sistemas de Información

Universidad Nacional de Lujan

Febrero, 2014

Palabras Clave

Ingeniería de requerimientos, Requerimientos, Volatilidad de requerimientos, Software, Requerimientos de cambio, metodologías de desarrollo

INDICE

CAPITULO I

I 1 – INTRODUCCIÓN

La Ingeniería de Requerimientos es una de las etapas fundamentales de la Ingeniería de Software y un desafío importante al momento de comenzar un proyecto de desarrollo de software y no por ser la primera del proceso de desarrollo sino porque una de las tareas más difíciles a la hora de desarrollar software es comprender que características y necesidades proporcionara el mismo, y en gran medida es debido a la incidencia que tiene la parte humana. (Pressman, 2006)

Esta etapa como en ninguna otra a lo largo de proceso de desarrollo se basa en la relación entre el cliente y usuario del futuro producto de software con el equipo que llevara a cabo la construcción del mismo. A medida que avanza esta comunicación se desprenden las características del sistema, indicando en detalle que espera el usuario en cuanto a funcionalidad como apariencia estética. Otro resultado de esta relación de ida y vuelta es determinar cuáles de estas características el usuario considera excluyente hasta el punto que sin ellas el sistema no tendría razón de ser, así como también las restricciones del sistema y las características que quedaran fuera del desarrollo.

La información definida junto al cliente/usuario sirve de base para poder realizar una estimación del esfuerzo que insumirá el desarrollo de cada una de las características del sistema, junto con sus correspondientes costos. Acto seguido se podrá planificar un cronograma para monitorear el desarrollo.

Por todo lo mencionado anteriormente, la ingeniería de requerimientos resulta de vital importancia para el éxito de un proyecto de desarrollo de Software. Es decir aunque en esta etapa se intentara obtener un conjunto de requerimientos válidos, existen diversas causas que pueden provocar la necesidad de que estos requerimientos cambien y es también tarea de la Ingeniería de requerimientos identificar esas posibles casusas para realizar una adecuada gestión de tales cambios, para aumentar la probabilidad de éxito del proyecto de desarrollo de software. (Nurmuliani, 2007)

Se denomina volatilidad de requerimientos a los cambios que a lo largo del ciclo de vida de un proyecto de desarrollo de software puedan surgir a los requerimientos definidos inicialmente. Cabe destacar que estos cambios pueden ocurrir en cualquiera de las etapas del proyecto. (Singh, 2012)

I 2 – OBJETIVOS

El objetivo del trabajo propuesto es realizar una investigación teórica, relevando las actuales metodologías de desarrollo de software, con el fin de profundizar en cómo se realiza la gestión de los requerimientos y que rol cumple el cliente o futuro usuario en este proceso. Del análisis de las metodologías se pretende encontrar:

- Cuáles son las prácticas que generan Volatilidad en los requerimientos.

- En qué medida afecta la volatilidad de los requerimientos el éxito de un proyecto de desarrollo de software.
- Reconocer como manejan las distintas metodologías la volatilidad de requerimientos.

Como resultado del análisis mencionado se pretende reconocer que características de las prácticas analizadas pueden ser útiles para intentar reducir en la medida de lo posible la volatilidad de requerimientos y mejorar la probabilidad de finalizar un proyecto exitosamente.

I 3 – JUSTIFICACIÓN

La volatilidad de requerimientos es uno de los principales riesgos para que un proyecto de desarrollo de software no tenga un cierre exitoso y en el peor de los casos no llegue a término. (Mundlamuri, 2005)

La principal consecuencia de la volatilidad de requerimientos es que genera trabajo adicional. Es decir al detectar que un requerimiento no tiene las características solicitadas por el cliente/Usuario o no satisface las necesidades planteadas habrá que destinar recursos para realizar un nuevo análisis, nuevos diseños su consecuente codificación extra. Este trabajo adicional generara desvíos en la planificación original del proyecto, incrementando los costos del mismo.

En este punto se considera oportuno mencionar, que surge un aspecto que impacta de manera directa en el proyecto de desarrollo de software, porque se puede optar por no realizar los cambios solicitados a los requerimientos, con el afán de cumplir con la planificación original, pero en este caso estaríamos entregando al usuario un producto que no le sería útil. Y si atendemos todos los cambios solicitados al sistema, el proyecto puede convertirse en no rentable.

Es de vital importancia entonces encontrar en base a las técnicas existentes para el desarrollo de software una óptima utilización de las mismas para que de ser posible minimicen el impacto de la volatilidad de requerimientos.

I 3.1 - Estructura del documento.

El presente documento se basara en la Ingeniería de requerimientos y como las metodologías actuales de desarrollo de software gestionan los cambios en los requerimientos. El mismo se organizara en cinco capítulos:

Capítulo I: Introduce el tema, remarcando los objetivos y justificación del trabajo.

Capítulo II: Presenta la Ingeniería de requerimientos, junto con sus etapas e importancia en el proceso de desarrollo de software.

Capítulo III: Introduce el concepto y causas de la volatilidad de requerimientos, remarcando cómo impacta en los distintos aspectos de un proyecto según cada fase del proceso de desarrollo de software.

Capítulo IV: Se realizara un repaso acerca de las distintas metodologías de desarrollo de software, haciendo énfasis en cómo cada una de ellas realiza la gestión de los requerimientos.

Capítulo V: Conclusiones Generales del trabajo.

CAPITULO II

II 1 - INGENIERIA DE SOFTWARE

En la actualidad la mayoría de las empresas dependen en gran medida del software que controla sus sistemas. Esto provoca que el éxito o fracaso de proyectos de desarrollo de software tenga un alto impacto en la economía y funcionamiento de estas empresas. Esta problemática evidenciada hace varios años se conoció como "Crisis de Software". La solución fue equiparar la producción del software con la de otros productos, generando la necesidad de tratar el software de manera ingenieril para, de esta manera, poder tener un control exhaustivo de todo el proceso de desarrollo de software, de manera que de tener visibilidad de los posibles inconvenientes que puedan surgir en el proceso y realizar una administración adecuada para tener una alta probabilidad de éxito del proyecto.

Aunque varios autores han definido Ingeniería de software, nosotros nos quedaremos con la siguiente:

"La Ingeniería de Software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este después que se utiliza." (Sommerville, 2011)

Dado lo complejo de la Ingeniería de software, debido a las variadas actividades que comprende se justifica la necesidad de otras disciplinas más específicas. En este trabajo nos centraremos en la Ingeniería de Requerimientos y su problemática.

II 2 - INGENIERIA DE REQUERIMIENTOS

La ingeniería de requerimientos se centra en el punto de partida para cualquier proceso de desarrollo de un producto de software. Nos permite decidir que necesidades planteadas por los clientes/usuarios serán satisfechas, evaluando aquellas características que aunque deseables por el cliente/usuario sean necesarias para que el software tenga una razón de existencia, que las mismas sean viables de realizar, así como también las propiedades que quedaran fuera de alcance del desarrollo.

A continuación se presentan algunas definiciones de Ingeniería de Requerimientos:

"Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software". (Pressman, 2006)

"La ingeniería de requerimientos es el proceso de desarrollar una especificación de software. Las especificaciones pretender comunicar las necesidades del sistema del cliente a los desarrolladores del sistema". (Sommerville, 2011)

Debido a la heterogeneidad de puntos de vista, gustos y prejuicios de los distintos usuarios que participan en el proceso de determinación de requerimientos, en esta etapa aparecen problemas que deben solucionarse tratando de llegar a acuerdos mediante la negociación entre los diferentes actores. Desde este punto de vista, la ingeniería de requerimientos es un proceso socio-técnico.

Esta etapa tiene la particularidad de ser determinante para resto del proceso de desarrollo de software, es decir los errores que tengan su origen en la misma y no sean detectados a tiempo serán motivo de problemas en alguna de las etapas siguientes, con el agravante que cuanto más avanzado en el proceso de desarrollo se visualice el problema más costoso será solucionarlo, hasta el punto extremo que el cliente/usuario no reciba realmente el software que cumpla con sus necesidades. (Mizuno, 1983)

II 2.1 – Importancia de la Ingeniería de Requerimientos

En el documento "Ingeniería de requerimientos" la autora Lizka Johany Herrera enumera las bondades de esta técnica: (Lizka, 2003)

- "Permite Gestionar las necesidades del proyecto en forma estructurada"
- "Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados"
- "Disminuye los costos y retrasos del proyecto"
- "Mejora la calidad del software"
- "Mejora la comunicación entre equipos"
- "Evita rechazos de usuarios finales"

II 2.2 - Fases de la Ingeniería de Requerimientos

La ingeniería de requerimientos está conformada de cuatro fases principales:

II 2.2.a - Estudio de viabilidad:

Con el objetivo de determinar si el proyecto seguirá adelante, se realiza un análisis para detectar los siguientes factores:

- Existencia de tecnología para satisfacer las necesidades planteadas
- Si el Software será rentable para la empresa que sugiere su necesidad.
- Si su desarrollo se enmarca dentro de los recursos económicos y de tiempos que la empresa tiene presupuestados.

II 2.2.b - Obtención y análisis de requerimientos:

El Objetivo de esta fase es obtener las necesidades que debe satisfacer el software. Valiéndose de entrevistas con los posibles usuarios del futuro software, observación directa de las tareas del mismo y análisis de productos de software existentes, se genera la documentación necesaria que servirá de base a la siguiente fase de especificación de requerimientos.

II 2.2.c - Especificación de requerimientos:

El objetivo de esta fase es traducir formalmente las necesidades detectadas en la etapa anterior en un conjunto de requerimientos. Dado que en esta fase es donde tiene su origen cada uno de los requerimientos del software, se considera oportuno introducir su definición:

> "Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste". (Sommerville, 2011)

Es decir un requerimiento es la descripción de una condición o característica que debe cumplir el software, cuyo origen es una necesidad detectada por el futuro usuario.

II 2.2.c.1 - Clasificación de requerimientos: Se puede realizar una clasificación de los requerimientos en base a sus características:

- Ámbito
  - *Software:* Significa que el requerimiento debe ser cumplimentado a nivel de software.
  - *Hardware:* Significa que el requerimiento debe ser cumplimentado a nivel de hardware.
  - *Sistema:* Significa que el requerimiento debe ser cumplimentado a nivel de sistema, es decir hardware y Software.

- Característica
  - *Funcional:* Definen en detalle las funciones que debe realizar el software, describiendo las salidas que el Software obtendrá para determinadas entradas, como también debiera comportase con determinadas excepciones. Es importante destacar que un requerimiento funcional hace referencia a que debe realizar el Software pero sin entrar en el detalle de cómo lo realizara. En ocasiones se incluyen entre los requerimientos funcionales las funciones que el software no debe realizar.
  - *No Funcional:* Definen las características que especifican o limitan en mayor o menor medida la disponibilidad y operación del sistema y su desarrollo. Los requerimientos no funcionales también tienen su origen en necesidades del usuario, es decir pueden desprenderse de solicitudes de operación del Sistema; o

por necesidades de la organización en cuanto a temas presupuestarios, políticos o de infraestructura de la organización, de interconexión con otros sistemas o de seguridad y estándares requeridos.

El siguiente Cuadro muestra los tipos de requerimientos no funcionales que se pueden distinguir: (Sommerville, 2011)

Tabla 1 – Tipos de Requerimientos no funcionales

Requerimiento de producto	Requerimiento de Usabilidad Requerimiento de Eficiencia Requerimiento de Fiabilidad - Requerimiento de Eficiencia - Requerimiento de espacio Requerimiento Portabilidad
Requerimiento Organizacionales	Requerimiento de Entrega Requerimiento de implementación Requerimiento de estándares
Requerimientos Externos	Requerimiento de Interoperabilidad Requerimiento éticos Requerimiento legislativos - Requerimiento de privacidad - Requerimiento de Seguridad

- Audiencia: Indica que tipo de personas tienen que entender el requerimiento.
  - *Usuarios:* Generalmente se definen en un lenguaje natural, con el menor contenido de componentes técnicos posible.
  - *Desarrolladores:* Generalmente se definen mediante modelos y un lenguaje técnico (por ejemplo; técnicas estructuradas y/u orientadas a Objetos)

II 2.2.d - Validación de requerimientos:

El objetivo de esta etapa es validar que la especificación de requerimientos generada en la etapa anterior sea correcta. Esta etapa tiene vital importancia ya que es el momento de detección y corrección de errores a los requerimientos detectados.

Para la validación de los requerimientos se tiene en cuenta una serie de propiedades que debería reunir una especificación de requerimientos. Estas se enumeran a continuación:

- *Comprensible:* Por todo lo mencionado anteriormente respecto a los problemas de la ingeniería de requerimientos en relación a la parte social y comunicacional entre usuarios y desarrolladores, es recomendable tener especial cuidado en este aspecto de un

requerimiento. Ya que es imprescindible que la especificación sea comprensible por ambas partes puede pensarse en formas diferentes de redacción para cada involucrado.

- *No ambiguo:* Todos los requerimientos presentes en la especificación deben tener una sola interpretación.
- *Completo:* La especificación debe incluir todas las funciones, restricciones y transformaciones que el usuario solicito como necesidad, es decir que el usuario al verificar la especificación no reconoce ninguna de sus propuestas como ausente.
- *Consistente:* Todo requerimiento no debe entrar en contradicción con otro requerimiento presente en la especificación.
- *Verificable:* Indica que debe existir la menos un mecanismo de prueba para corroborar que el sistema final cumple con el objetivo del requerimiento planteado.
- *Real:* De debe verificar que los requerimientos puedan ser implementados en la actualidad del desarrollo, es decir se deben evaluar aspectos como por ejemplo tecnológicos o de presupuesto que determinen viabilidad de los mismos.
- *Modificable:* Un requerimiento debe permitir ser modificable de una manera sencilla manteniendo todas las características deseables acerca del mismo.
- *Rastreable:* Esta es una característica necesaria para el mantenimiento de la especificación a medida que el proceso de desarrollo avanza. Es decir es deseable que se pueda referenciarse como origen de documentos generados en etapas posteriores. Una buena práctica es que se pueda identificar unívocamente a cada requerimiento mediante algún código.

En la práctica, para grandes sistemas, se dificulta cumplir con las propiedades enumeradas. Alguna razón de esto es que los clientes a menudo suelen omitir detalles importantes de su operatoria habitual o tienen la dificultad de relacionar sus necesidades con las especificaciones generadas, pueden tener necesidades que generen contradicciones que no sean fácilmente reconocibles incluso después de un análisis exhaustivo. Con la premisa que los requerimientos sean comprensibles se cae inconscientemente en ambigüedades. En cuanto a los requerimientos no funcionales los clientes se encuentran generalmente con la imposibilidad de proporcionar una información cuantitativa de las metas que debe cumplir el sistema. (Sommerville, 2011)

Cabe destacar que si bien el objetivo de la Ingeniería de requerimientos es generar un documento verificado y confiable con todos los requerimientos que se desean del software, su función no tiene como límite solo el inicio, sino que continúa a lo largo de todo el ciclo de vida del proceso de desarrollo de software.

El objetivo de esto es realizar un mantenimiento de los requerimientos y de esta manera verificar que el software construido se corresponde con la especificación generada, es decir esto tiene que ver con la trazabilidad y verificabilidad de los requerimientos. Otra razón importante es que existe la posibilidad

de que algunos requerimientos cambien a lo largo del proceso de desarrollo, motivo por el cual se debe realizar una gestión de los cambios que puedan ocurrir. (Sawyer y Kontoya 1999)

CAPITULO III

III 1 – VOLATILIDAD DE REQUERIMIENTOS

Una vez que se han identificado todas las necesidades del cliente, y estas necesidades dieron origen a los requerimientos, los mismos serán documentados con el objetivo de celebrar un acuerdo con el cliente acerca de que requerimientos serán desarrollados, este documento es la especificación de requerimientos de software y está conformada por los requerimientos básicos del sistema. Aunque la intención de esto es intentar obtener y documentar todos los requerimientos en la etapa inicial del proyecto, existen diversos factores que provocan que inevitablemente los requerimientos puedan sufrir alguna modificación a lo largo del ciclo de vida del proyecto de desarrollo de software. (Barry, 2002) El reporte CHAOS de 1998 indica que si estos cambios ocurren en reiteradas ocasiones a lo largo del proyecto, pueden provocar una incertidumbre importante en el mismo, siendo uno de los principales factores que provocan que un proyecto no sea exitoso. (TSG, 1998)

III 1.1 – Requerimiento de cambio

Al presentarse alguna modificación a los requerimientos documentados en la especificación de requerimientos de software se conoce como requerimientos de cambio y se sigue un proceso que comienza con una solicitud de cambio. El objetivo de la utilización de un proceso para realizar una administración de los cambios solicitados a los requerimientos es tener un control de los cambios que se han realizado a los requerimientos básicos.

El proceso de gestión de cambios se conforma por las siguientes etapas: (Somerville, 2011)

- Analizar el problema, Especificar el cambio: Se realiza un análisis del problema planteado para verificar su validez y existencia, a fin de poder realizar una especificación que pueda ser presentada a quien solicito el cambio para su aprobación.
- Analizar el cambio, estimar las modificaciones necesarias: Se realiza un análisis de los cambios necesarios, basándose en la información que se posee acerca del resto de los requerimientos, con el objetivo de poder estimar el costo y recursos necesarios para las modificaciones solicitadas. Con esta información se analiza si se llevan a cabo las modificaciones o se dejan sin efecto.
- Modificación de especificación de requerimientos, y su aplicación al sistema: Es importante modificar el documento de requerimientos básicos del sistema antes de implementar los cambios con el objetivo de que el mismo quede alineado con el software desarrollado.

Los cambios a los requerimientos pueden ser de tres tipos diferentes:

- Requerimientos agregados: Hacen referencia a necesidades identificadas después de que se han acordado los requerimientos básicos del sistema y por lo tanto no figuran en el documento correspondiente.
- Requerimientos Eliminados: Hacen referencia a necesidades que figuran en el documentos de requerimientos básicos pero por alguna razón no son necesarias y se dejan sin efecto.
- Requerimientos Modificados: Hacen referencia a necesidades que no se han entendido de manera correcta al acordar los requerimientos básicos del sistema, o que el usuario al utilizar el software se percata que necesitaba tal requerimiento pero con modificaciones.

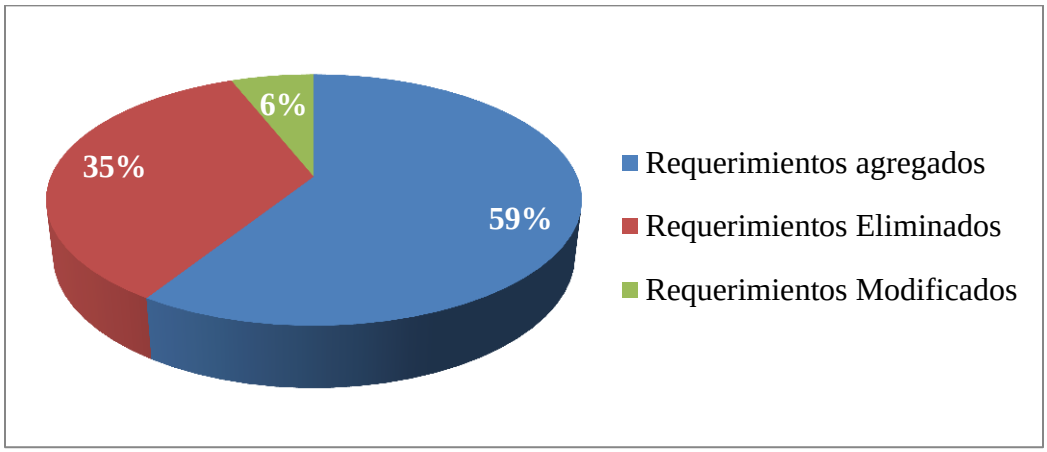

Grafico 1 – Tipos de requerimientos de cambio. (Stark, 1999)

La volatilidad de requerimientos puede verse como los cambios en las necesidades de los clientes a lo largo del ciclo de vida de un proyecto de desarrollo de software. Como no existe una definición universal, varios autores han definido el concepto en sus investigaciones:

"Se describe a la Volatilidad de requerimientos como la diferencia en la información necesaria para identificar las necesidades de los usuarios y la información que poseen los desarrolladores" (Nidumolu, 1996)

"Se define Volatilidad de requerimientos como la tendencia de los requerimientos para cambiar a través del tiempo en respuesta a las necesidades cambiantes de los clientes, partes interesadas, la organización y el entorno de trabajo." (Nurmuliani, 2007)

"Los cambios en los requerimientos, ya sea modificaciones, nuevos requerimientos o requerimientos eliminados - después de que el conjunto básico ha sido aceptado por los clientes y los desarrolladores de software – es lo que se conoce como volatilidad de requerimientos."(Stark, 1999)

Se denomina volatilidad de requerimientos a los cambios que a lo largo del ciclo de vida de un proyecto de desarrollo de software puedan surgir a los requerimientos definidos inicialmente. Cabe destacar que estos cambios pueden ocurrir en cualquiera de las etapas del proyecto. (Singh, 2012)

Así como Singh menciona que la volatilidad de requerimientos puede ocurrir a lo largo de todo el ciclo de vida del proyecto desarrollo de software, Javed (Javed, 2004) hace una diferenciación de la volatilidad de requerimientos en base a la etapa de proyecto en la cual se solicitan tales cambios.

– *Pre-Release:* Los cambios en los requerimientos ocurren antes de que el producto esté disponible para el cliente. En este punto vale hacer también una distinción:

- *Pre-Especificación:* Los cambios son solicitados cuando se está en la etapa de definición de las necesidades del cliente, en el momento de elaboración de la especificación, en plena negociación entre el cliente y los desarrolladores, es decir todavía no ha sido documentada ni acordada la especificación de requerimientos. Es importante destacar que si estos cambios son realizados en forma correcta, se le dará un valor agregado al producto de software.

- *Post-Especificación:* Los cambios son solicitados una vez que se ha documentado y acordado la especificación de requerimientos, es decir pueden ocurrir en las etapas posteriores del proceso de desarrollo de software, ya sea desde el diseño hasta las pruebas del producto de software.

– *Post-Release:* Los cambios en los requerimientos ocurren luego que el producto se ha instalado, y está disponible para el uso por parte del cliente.

Los cambios que se soliciten luego de que se ha documentado y acordado la especificación de requerimientos, pueden afectar económicamente el proceso de desarrollo como así también el cronograma y calidad del producto de software ya que existe la posibilidad de insertar defectos en el mismo. (Javed, 2004)

Durante el proceso de desarrollo software se realizan sobre el proyecto mediciones, con el objetivo de detectar el estado en el que se encuentra cada una de las fases del proyecto y de ser necesario aplicar las correcciones correspondientes. El caso de la volatilidad de requerimientos no es la excepción, se realiza una recolección de datos con el objetivo de tener visibilidad del impacto que pueda tener en el proyecto los cambios solicitados por las partes interesadas. Entre los datos a recolectar se encuentran: (Mundlamuri, 2005)

- Cantidad Total de requerimientos
- Cantidad y Tipos de necesidades: Indica la cantidad de cambios de Eliminación, modificación o Adición de requerimientos
- Cantidad de cambios realizados en la etapa de negociación de la especificación de requerimientos y cantidad de cambios una vez que el software está a disposición del cliente.

- Esfuerzo medido en tiempo real y planificado para llevar a cabo cada requerimiento de cambio.

En base a métricas recolectadas en su investigación "An Examination of the Effects of Requirements Changes on Software Maintenance Releases", George Stark (Stark, 1999) definió una ecuación para calcular volatilidad de requerimientos de un proyecto:

VR = ((A + E + M) / RT) * 100 (1) (Stark, 1999)

Donde VR, Volatilidad de requerimientos; A, Cantidad de requerimientos agregados; E, Cantidad de requerimientos Eliminados; M, Cantidad de requerimientos modificados; RT, Cantidad total de requerimientos en la especificación acordada entre las partes interesadas.

III 1.2 – Causas

La principal causa de volatilidad de requerimientos se atribuye a que el software es un proceso que es dinámico por naturaleza, lo que conduce a inevitables cambios en el conjunto de requerimientos iniciales. (Mundlamuri, 2005)

Sin embargo hay una serie de factores de los cuales varios de ellos no pueden ser evitados pero existen otros que tomando medidas al respecto se puede evitar que provoquen cambios a los requerimientos: (Singh, 2012)

- Factores Externos: Ingresan dentro del grupo de factores que no pueden ser manejados por ninguna de las partes interesadas y por lo tanto no pueden ser evitadas.
    - Regulaciones del Gobierno
    - Competencia de Mercado.
- Factores Internos
    - En el aspecto técnico
        - Limitaciones del producto
        - Falta de experiencia de los integrantes del equipo de desarrollo
        - Retroalimentación de otras fases del ciclo de vida del proceso de desarrollo
        - Tamaño de proyecto y Gran cantidad de requerimientos
        - Variaciones en el costo de herramientas de software y hardware
    - Cambios en el entorno empresarial
    - Cambios constantes en los requerimientos que el usuario solicita,
    - Diversidad en las necesidades de los futuros usuarios del producto software: Para prevenir que este factor genere cambios en los requerimientos, es aconsejable que en la etapa inicial se identifique a los usuarios más significativos del futuro producto de

software, una vez realizado esto darle a cada grupo identificado el espacio adecuando para el planteo de sus necesidades.

- Carencia en la comunicación entre los futuros usuarios y el equipo de desarrollo del software.
- Irresponsabilidad del cliente.

Dentro de los factores evitables se puede incluir la comunicación entre las partes interesadas (futuros usuarios y equipo de desarrollo) y la retroalimentación de otras etapas del proceso de desarrollo.

III 1.3 – Impacto

La calidad de los requerimientos de un producto de software en una de las características que determinara el éxito o fracaso de un proyecto. Los constantes cambios a los requerimientos afectan esta característica considerada vital y por lo tanto tienen una consecuencia importante en el ciclo de vida del proceso de desarrollo de software, transformándose en un riesgo para el proyecto. (Hammer, 1998)

Al incorporar requerimientos a la especificación de requerimientos inicial o modificar o eliminar los existentes en ese grupo se incrementara el re trabajo y en consecuencia el costo de desarrollo y su consecuente desvío en la planificación inicial provocando un atraso en las fechas de liberación del software.

Los requerimientos de cambio afectan de manera significativa el normal desarrollo del ciclo de vida del proyecto cuanto más tarde en el proceso de presenten. Es decir, en la etapa de análisis de la especificación de requerimientos la volatilidad es alta pero como en la misma los requerimientos se encuentran en pleno estudio y se está decidiendo la duración del proyecto, como también los recursos asignados y la dimensión del sistema, la modificación de los requerimientos en esta etapa no tiene un gran impacto sobre el proyecto. (Zowghi, 2007)

Distinto es en las etapas subsiguientes, la etapa de diseño depende exclusivamente de los requerimientos y si estos tienen un alto grado de volatilidad la tarea de diseñar una solución y por consiguiente su planificación de desarrollo se verá dificultada. En cuanto a los cambios en la etapa de codificación cada uno de ellos tendrá su impacto en el código y si los mismos son concurrentes se puede dañar la estructura del programa y peor aún introducir errores. (Mundlamuri, 2005)

Los requerimientos de cambio en la etapa de pruebas indicaran que el producto de software no es realmente lo que el usuario espera. Esto afectara de manera crítica el proyecto, debido a que al darle curso al requerimiento de cambio se incurrirá inevitablemente en sobrepasar los plazos de entrega y los consecuentes costos asociados al re trabajo, y si no se incluyen las modificaciones solicitadas el software tendrá un impacto a nivel de calidad. (Mundlamuri, 2005)

III 1.3.1 - Impacto sobre el cronograma del proyecto

Al momento de comenzar un proyecto, la primera tarea a desarrollar es la creación de un cronograma, esto es debido a que si bien el ciclo de vida está conformado por varias tareas, desde el relevamiento de las necesidades de cliente hasta las prueba del producto de software, es necesario conocer cuando tendrá inicio y fin cada una de estas tareas. Un cronograma brinda la posibilidad de tener un control acerca de las tareas, de modo que cualquier atraso en alguna de ellas se manifieste claramente y permita la aplicación de las acciones adecuadas para continuar con el curso del proyecto. Generalmente es el mismo cliente que necesita saber en qué tiempos comenzara a tener disponibles entregables del proyecto. Esta planificación dará origen a los recursos necesarios para llevar a cabo el proyecto con sus consiguientes costos asociados.

Cuando el proceso normal de desarrollo se ve interrumpido por cambios en los requerimientos acordados, es muy probable que dichos cambios afecten las tareas de diseño, codificación o pruebas dependiendo en la etapa del proyecto en la cual se solicitan los cambios. En consecuencia muchas veces es necesario realizar una nueva planificación, ya que los plazos de entrega acordados al inicio se verán desfasados por los cambios solicitados. Un número importante de proyectos no han llegado a término por estos motivos, ya que en ocasiones los retrasos en la entrega del software hacen que el mismo no cumpla con su cometido, por ejemplo en estrategias de marketing. (Mundlamuri, 2005)

Es tarea de la gestión de requerimientos, trabajar junto al cliente para identificar cuáles de los requerimientos de cambio son excluyentes para las características deseables del software, a fin de dejar sin efecto o para etapas posteriores los requerimientos que no han sido identificados en este grupo y de esta manera no se modifiquen en gran medida los tiempos de entrega planificados inicialmente.

George Stark (Stark, 1999) definió una ecuación para calcular el porcentaje de atraso de un proyecto:

PAP = {1+ ((DR – DP) / (DP))} * 100 (1) (Stark, 1999)

Donde PAP, Porcentajes de atraso de proyecto; DR, cantidad de días Reales; DP, cantidad de días Planificados. Para que el proyecto haya cumplimentado la planificación inicial el resultado de la mencionada ecuación debería ser 100, Una valor superior a 100 indica que el proyecto ha incurrido en atrasos, mientras que un valor inferior a 100 indicara que el proyecto culmino antes de lo previsto.

III 1.3.2 - Impacto económico

El aspecto económico de un proyecto de desarrollo de software es uno de los factores fundamentales que hacen a la utilidad financiera del desarrollo de proyecto. El mismo será determinado por la duración del proyecto, así como también la cantidad y calidad de los recursos involucrados en el proyecto. Estos costos son determinados en el inicio del proceso de desarrollo mediante un acuerdo entre los clientes y la gerencia de la parte desarrolladora del software. Por lo tanto cuando se presentan

requerimientos de cambio las partes interesadas deben acordar un nuevo cronograma de entregas con sus correspondientes incrementos en los costos asociados. Es decir cuando se realiza una solicitud de cambio al planificar una nueva duración del proyecto, el mismo afecta directamente los costos de desarrollo y en este punto es donde clientes y desarrolladores evaluaran limitar el alcance de los requerimientos o extender los tiempos de entrega si es posible.

En ocasiones la parte desarrolladora de software valoriza los requerimientos con un valor bajo en las etapas iniciales del proyecto, esto es porque ante cambios en los requerimientos el normal proceso de desarrollo no se ve afectado en gran magnitud. Mientras que las modificaciones en instancias tardías del proceso de desarrollo se valorizan con un costo mayor, debido al impacto negativo que puede tener en el proyecto. (Singh, 2012)

III 1.3.3 - Impacto en calidad de producto

Muchas veces se menciona que calidad de un producto de software está relacionada con la no existencia de errores, pero un software solo esta momentáneamente libre de errores hasta que aparezca alguna falla. Y se deja de lado un factor más que importante al referirse a la calidad de un software, ese aspecto está ligado a la satisfacción del cliente/usuario del software, Es decir que todas las necesidades planteadas por el mismo hayan sido cubiertas, la entrega en los periodos de tiempo acordados y con los costos presupuestados inicialmente. (Ali Javeed, 2006)

En este punto es donde se introduce un conflicto, partiendo de la premisa que cuanto mayor será la volatilidad de requerimientos menor será la calidad del producto final. (Singh, 2012) Se explica en que por un lado al ser recurrentes los cambios realizados a lo largo del proceso de desarrollo de software, por ejemplo en la etapa de codificación, existe la posibilidad de inyectar errores en el software más allá del impacto cronológico y financiero que esto provoca. La otra faceta del conflicto es no darle curso a las solicitudes de requerimientos planteadas por el cliente/usuario, entonces sus necesidades no estarán cubiertas plenamente. Con ambas decisiones se le estaría restando calidad al proyecto de desarrollo, la primera provoca un software con probables errores y entregado fuera de término, y la segunda opción no cubre las necesidades planteadas por el cliente.

CAPITULO IV

IV 1 – VOLATILIDAD DE REQUERIMIENTOS EN LAS DISTINTAS METODOLOGIAS DE DESARROLLO

Los cambios en los requerimientos de software se presentan en gran medida porque el proceso de desarrollo es naturalmente dinámico, (Mundlamuri, 2005) y este dinamismo en el proceso, como en las organizaciones donde se llevan a cabo los desarrollos de software, provocan que existan una serie de factores causantes de tales cambios.

Los procesos de desarrollo han ido evolucionando introduciendo mejoras para la gestión de requerimientos y los desarrolladores son más conscientes que al comenzar un proyecto de desarrollo de software existen muchas probabilidades de que los requerimientos sufran cambios.

Por tal motivo el problema no son los requerimientos y sus cambios, sino que el inconveniente a solucionar es seleccionar el modelo de proceso adecuado que se seguirá durante el desarrollo de software, de manera que esos cambios puedan ser manejados y el impacto de los mismos en el proyecto se vea minimizado. (Javed, 2004)

En esta sección analizaremos los diferentes modelos de proceso utilizados en el desarrollo de software, resaltando como es su comportamiento frente a la problemática de la volatilidad en los requerimientos iniciales de un proyecto.

IV 2 – METODOLOGÍAS TRADICIONALES

IV 2.1 – Modelo en Cascada.

El modelo en cascada es un proceso derivado de otros procesos ingenieriles más generales, y es el modelo clásico utilizado por años en la industria del software. Representa las actividades fundamentales del proceso como etapas independientes.

Debe su nombre a la cascada que metafóricamente forma cada una de sus etapas con las posteriores, es decir cada una de las etapas subsiguientes utiliza la documentación revisada y aprobada en la etapa anterior. Por tanto hasta que una etapa no se encuentra finalizada, la etapa siguiente no puede dar comienzo. (Sommerville, 2011)

En el Grafico 2 se muestra una representación de este modelo.

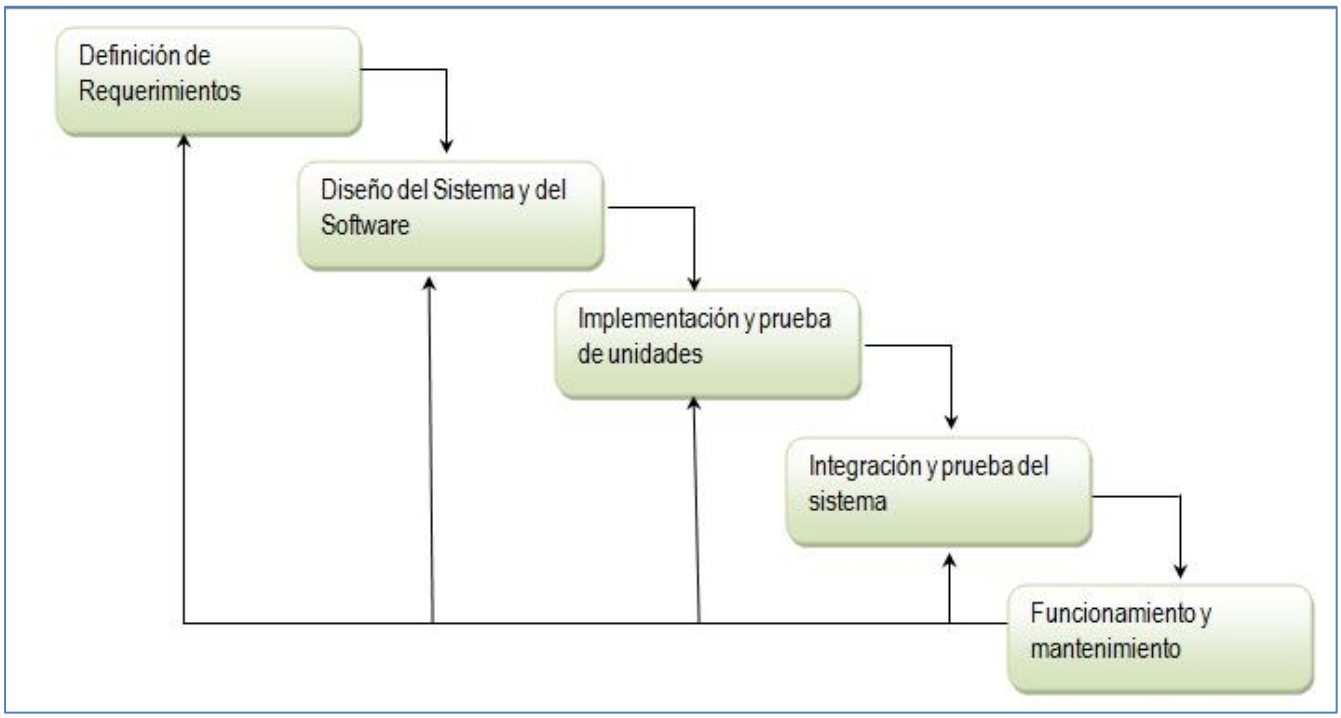

Grafico 2 – Modelo en Cascada. (Sommerville, 2011)

Este modelo requiere que el equipo de desarrolladores como primera medida identifique todos los requerimientos que describan la funcionalidad del futuro producto de software. Con estos requerimientos se creara un diseño que será utilizado durante todo el proceso de desarrollo. Cuando el equipo de desarrolladores da por finalizado el producto de software, el mismo es entregado al usuario final, y los problemas que puedan surgir de la utilización o modificaciones serán abordados en una etapa de mantenimiento. Este enfoque tiene la particularidad de que se interactúa con el usuario solo en la etapa de obtención de requerimientos, por lo tanto puede suceder que se esté entregando un producto que no es el que el usuario espera.

Referido a la volatilidad de los requerimientos, el gran problema que presenta el modelo en cascada, debido a su proceso secuencial, es que cualquier cambio en los requerimientos (requerimiento nuevo o modificado) retrotrae el proceso a la etapa de definición de requerimientos para su modificación y consiguiente rediseño. El informe del grupo Standish "Chaos Report" ejemplifica los problemas que tiene este modelo frente a la volatilidad de los requerimientos de un proyecto. Además destaca que, a pesar del uso generalizado del modelo en cascada, en esa época, había muy pocos proyectos exitosos.

El grafico 3 muestra esta estadística:

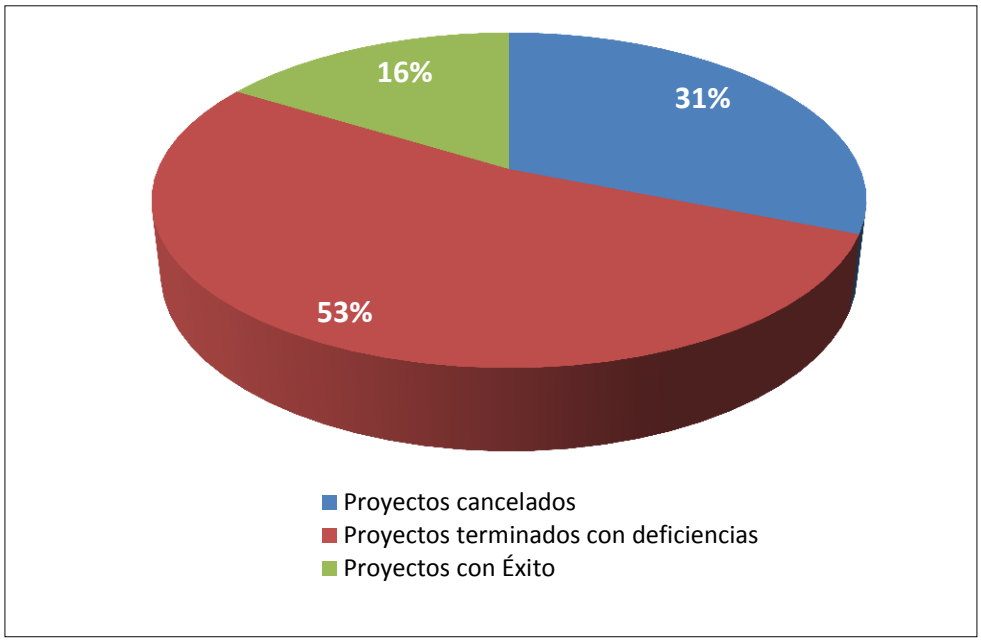

Grafico 3 – Chaos Report. (TSG, 1998)

Hubo intentos de extender el modelo en cascada. Un enfoque estuvo basado en prever los cambios que puedan ocurrir en los requerimientos de modo que la arquitectura del software permitiera la inclusión de los mismos sin afectar radicalmente, ni el proceso, ni el producto final. Esta aproximación tiene la limitación que no se tiene completa visibilidad de los posibles cambios. Otro enfoque consistió en introducir prototipos en la obtención de requerimientos de manera que al observar tales prototipos se pueda descubrir modificaciones a los requerimientos iniciales antes de ingresar en la etapa de diseño. Estos dos enfoques han contribuido a disminuir la volatilidad de requerimientos en el modelo en cascada. (Rajlich, 2006)

En resumen el modelo en cascada no es apropiado para proyectos con una alta probabilidad de que los requerimientos cambien. Si es eficiente para proyectos bien definidos en los cuales los requerimientos rara vez se modifican. (Mundlamuri, 2005)

IV 2.2 – Desarrollo evolutivo.

La idea de este modelo es desarrollar en base a los requerimientos obtenidos con el usuario una versión inicial del producto de software que será presentada y evaluada por el usuario con el objetivo de ir refinando la misma hasta obtener mediante iteraciones el producto de software esperado por el usuario. (Pressman, 2006)

En el Grafico 4 se muestra una representación de este modelo.

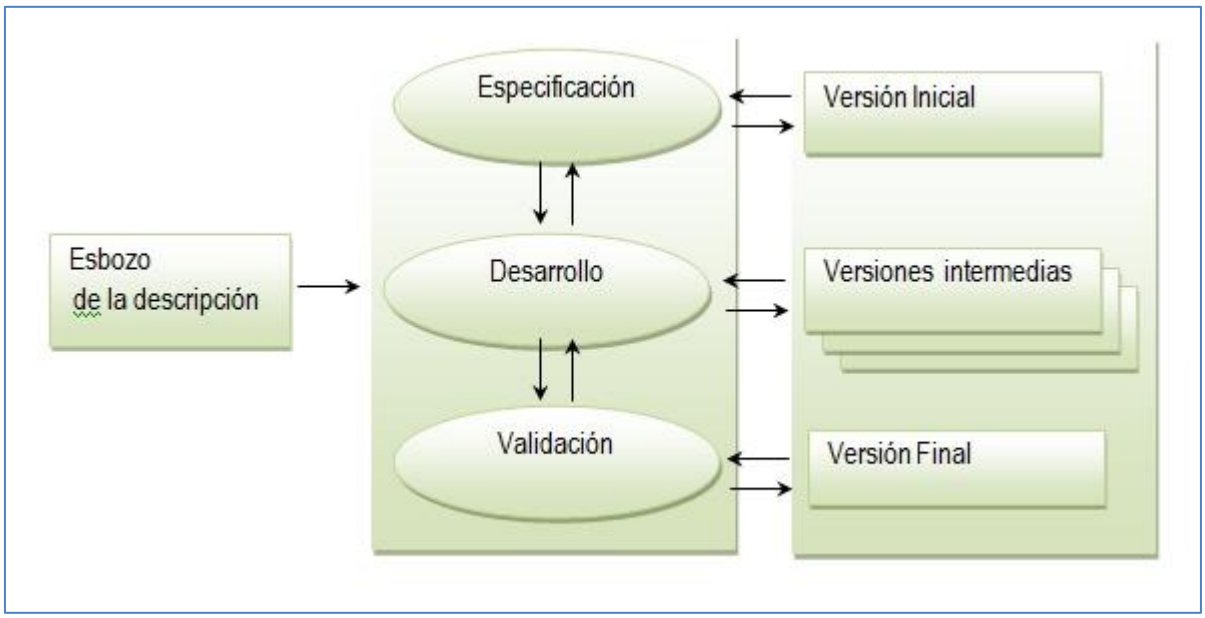

Grafico 4 – Modelo evolutivo. (Sommerville, 2011)

Tiene la ventaja sobre el modelo en cascada de que la interacción con el usuario es constante, tanto en el desarrollo exploratorio, donde se desarrollan los requerimientos comprendidos completamente para ir descubriendo junto a los usuarios los restantes pero basado sobre un producto de software, como en el desarrollo de prototipos.

Esta constante interacción con el usuario hace posible que se puedan detectar las necesidades del mismo a tiempo sin tener que esperar hasta el final del proceso como ocurre con el proceso en cascada.

Aunque este modelo resulta adecuado para proyectos con volatilidad en sus requerimientos se considera necesario mencionar los problemas de este modelo: la falta de visibilidad debido a que al no conocerse todos los requerimientos del productos de software no se tiene la certeza de los tiempos de finalización del proceso, el alto costo de generar documentación para cada iteración y la deficiente estructura que tiene el producto de software debido a los constantes cambios. Por este motivo este modelo de proceso es adecuando para sistemas hasta un tamaño medio. (Sommerville, 2011)

IV 2.3 – Modelo en espiral.

El modelo en espiral es un modelo de proceso del tipo evolutivo, que combina la práctica de realizar iteraciones de prototipos con las características de control del modelo en cascada. Es decir el proceso de desarrollo de software consiste en realizar entregas que pueden ser un modelo o un prototipo que a medida que se avanza con las iteraciones constituyen versiones más completas del producto de software. Este modelo de desarrollo le brinda una considerable importancia a los riesgos que puedan aparecer en el proceso. (Pressman, 2006)

En el Grafico 5 se muestra una representación de este modelo.

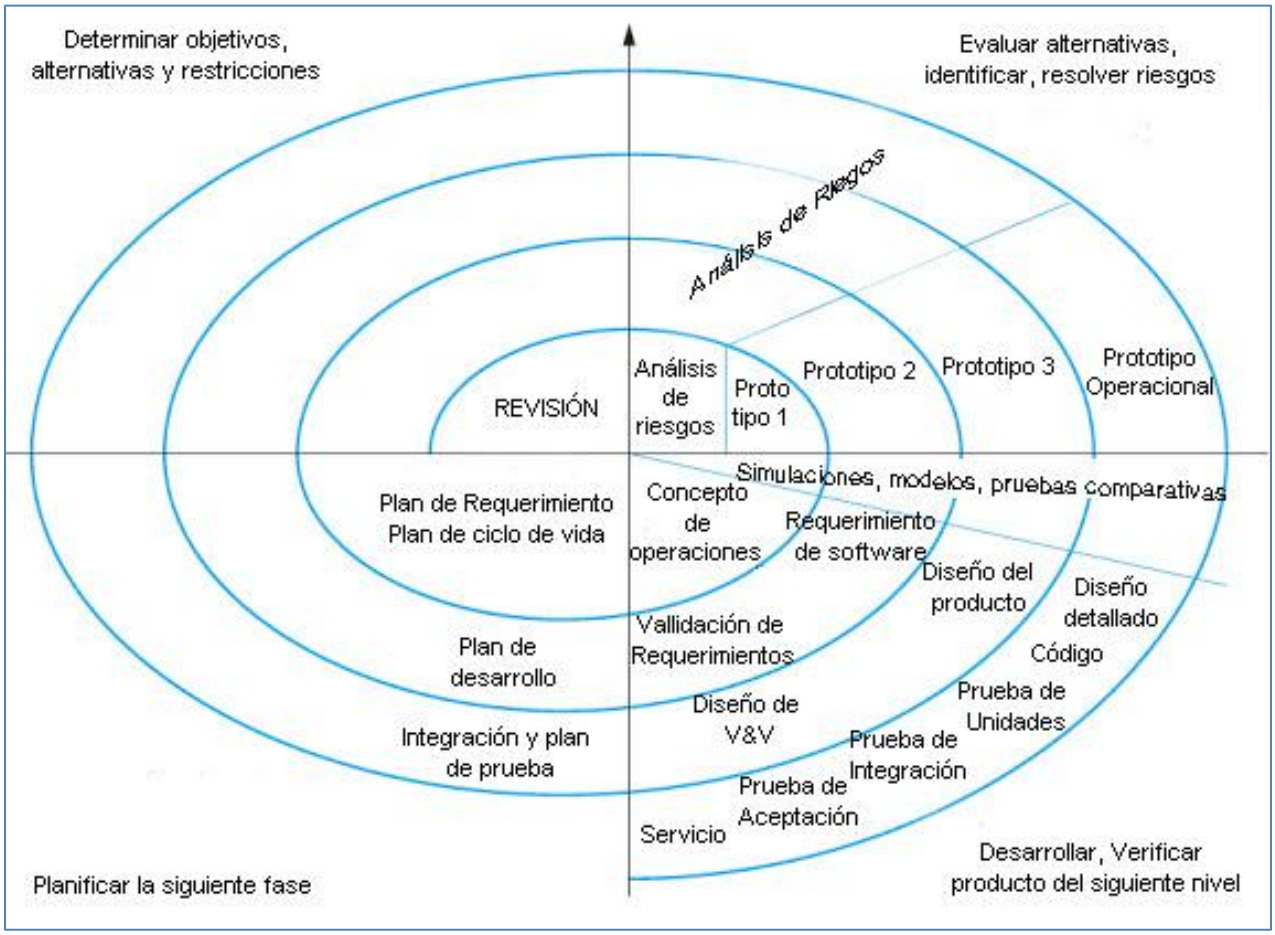

Grafico 5 – Modelo en espiral. (Sommerville, 2011)

Cada ciclo de la espiral presenta una fase del proceso de desarrollo de software.

Un ciclo comienza con la obtención del grupo de los requerimientos iniciales que serán desarrollados en la primera iteración. Acto seguido se realiza un análisis de las posibles alternativas de solución cada una de las cuales serán evaluadas a fin de analizar los riesgos que implica cada una de ellas y cómo resolverlos. Como resultado de este análisis se optara por la alternativa que tiene el riesgo más asumible, la cual se procederá a desarrollar. Si el usuario solicita modificaciones en el producto de software se comienza un nuevo ciclo con un nuevo análisis de alternativas y sus consiguientes riesgos. Los ciclos continuaran sucediéndose hasta que el producto de software no necesite ningún nuevo cambio.

Por su naturaleza iterativa este modelo de proceso es adecuado para proyectos en los cuales los requerimientos tienden a cambiar. Si se agregan nuevos requerimientos al grupo inicial bastara con añadir un nuevo ciclo al espiral para incorporarlo.

IV 3 – METODOLOGÍAS AGILES

Los procesos agiles tuvieron su origen en la necesidad de solucionar los problemas con los que un equipo de desarrollo se encontraba al utilizar las metodologías tradicionales de desarrollo de software. Esta metodología tiene sus bases en los siguientes lineamientos que pretende solucionar: (Pressman, 2006)

- La dificultad para determinar los requerimientos que cambiarían, como cuáles serán prioritarios para el cliente al promediar el proyecto.
- Sostiene que el diseño y desarrollo deben ejecutarse en forma simultánea.
- La dificultad de planificar la totalidad de las tareas de análisis, diseño, desarrollo y pruebas.

Un proceso ágil es naturalmente un proceso iterativo para cada una de las tareas comunes en el desarrollo de software (especificación, desarrollo y entrega) basándose en entregas incrementales, que el cliente tiene la posibilidad de evaluar. Fue diseñado para solventar la problemática del cambio en los requerimientos iniciales de un proyecto de desarrollo de software, lo cual no significa que los requerimientos iniciales no puedan cambiar, sino que es un enfoque para hacer frente a esos cambios y que tanto el proyecto como la calidad del producto de software no se vean afectados. (Sommerville, 2011)

En la actualidad existen varios métodos que comparten la misma filosofía siguiendo una serie de principios que se resumen en la Tabla 2.

Tabla 2 – Principios de los métodos ágiles. (Sommerville, 2011)

Principio	Descripción
Participación del cliente y entrega incremental	Los clientes tienen una activa participación desde el inicio al final del desarrollo. Al final de cada iteración tendrá el rol de evaluar cada entrega, y participar en la obtención de nuevos requerimientos y prioridades para la iteración subsiguiente.
Orientado a personas	Se libera al equipo de desarrollo para seguir su metodología de trabajo sin apegarse a procesos formales.
Flexibilidad ante el cambio	Como es una metodología que tiene su origen en el cambio de los requerimientos, la arquitectura del producto de software debe estar diseñada para aceptar de una forma sencilla tales cambios sin resentir la estructura del mismo, ni afectar la calidad del producto.
Simplicidad	En la medida de lo posible se trabajara en eliminar la complejidad del sistema

Uno de los principios fundamentales de las metodologías agiles es la injerencia que tiene el usuario en la toma de decisiones en cuanto a los requerimientos que serán incluidos en cada iteración, así como también la validación de las entregas.

Una herramienta para que el usuario, a veces inexperto, pueda reconocer las necesidades planteadas en un modo visible son los prototipos. En la actualidad va tomando mayor importancia una técnica, que si bien no será desarrollada en este trabajo vale la pena mencionar, para la realización de prototipos basados en la experiencia de usuario y el grado de usabilidad que el mismo percibe de la aplicación. Esta técnica tiene sus bases en diseñar cada prototipo desde el punto de vista del usuario particular, tal diseño tendrá el nivel de detalle dictaminado por la realidad del proyecto en el que está inmerso. Como principal beneficio de la técnica de prototipos basado en la usabilidad es que como el diseño es iterativo y al tener como prioridad las necesidades del usuario se asegura que al finalizar el producto de software este cumple realmente con las expectativas del cliente. (Fagalde, 2011)

Todas las metodologías de desarrollo tienen sus límites y los procesos agiles no son la excepción. Esta metodología por su naturaleza no es apropiada para desarrollos a gran escala donde las partes involucradas en el proceso distan geográficamente ni para proyectos que requieran un alto grado de documentación de los requerimientos.

IV 3.1 – Programación extrema

Dentro de las metodologías agiles la programación extrema representa quizás el enfoque más utilizado. Hace hincapié en eliminar del proceso actividades consideradas improductivas como ser la planificación que de surgir cambios en los requerimientos sufrirá un desfasaje y/o la obligación de una re planificación. La premisa de este método se fundamenta en la comunicación entre el equipo desarrollador y el usuario, simplicidad en el diseño de la arquitectura lograda sin considerar cambios futuros y retroalimentación resultante de las entregas y evaluaciones por parte de los usuarios hacia el equipo desarrollador. (Pressman, 2006)

Los requerimientos son presentados como escenarios construidos por la pareja de desarrollador y usuario. Estos escenarios incluyen las pruebas que se realizaran una vez terminado el desarrollo de la iteración, sin dar por finalizada dicha iteración hasta que tales pruebas no sean exitosas y aceptadas por el usuario.

En el Grafico 6 se muestra una representación del modelo de programación extrema.

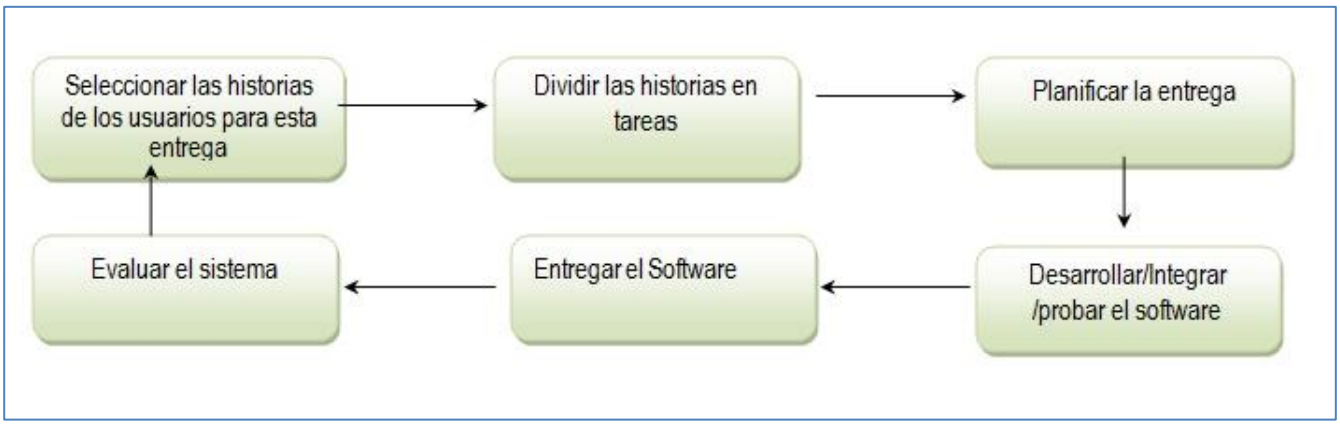

Grafico 6 – Programación Extrema. (Sommerville, 2011)

La programación extrema como toda metodología ágil está diseñada para adaptarse a los constantes cambios en los requerimientos. Una problemática de los cambios constantes radica en que pueden dañar la estructura del producto de software. La programación extrema hace frente a este problema con el principio de refactorización, es decir antes de introducir un cambio se deberá evaluar si la arquitectura de software se verá afectada como conjunto y de ser así se deberá rediseñarse de manera que el cambio pueda ser incorporado de un modo sencillo al sistema. Cabe destacar que esta refactorización debe ser realizada en forma constante para así mantener la baja complejidad de la solución implementada. *(Sommerville, 2011)*

CONCLUSIÓN

Durante el capítulo III se ha explicado que es la volatilidad de requerimientos y como esto puede afectar seriamente a los proyectos de software. Se mencionan algunas investigaciones que miden el impacto de la volatilidad de los requerimientos.

Se ha tipificado los tipos de cambios en Requerimientos Agregados, Requerimientos Eliminados y Requerimientos Modificados.

También se ha tipificado la etapa del proceso de desarrollo en el cual estos cambios pueden aparecer en Pre-Release en etapa de especificación, Pre-Release en etapa de construcción y Post-Release.

Dadas estas tipificaciones, se cree que es factible ponderar empíricamente el impacto de la volatilidad de requerimientos en los proyectos de software.

Sin importar el tipo de cambio y en etapa de especificación, el impacto del cambio es relativamente bajo, pero a medida que avanza el proceso de desarrollo, el impacto va a depender fuertemente del aspecto del software que se quiera alterar (ya sea para agregar, modificar o quitar características) y el grado de avance en la implementación de los requerimientos dependientes de estos. En etapa de Post-Release, el impacto puede ser alto pero una estrategia válida para superarlo puede ser tratar los cambios como un proyecto nuevo de evolución del producto, con nuevos plazos y nuevo presupuesto.

Si el cambio propuesto altera la estructura planteada del software y el proyecto ya está en fase de desarrollo, el impacto será alto en varios aspectos: Calidad, Fiabilidad, Costos, Plazos de entrega, etc.

Dicho esto, la hipótesis a trabajar es que la volatilidad de requisitos tiene un mayor impacto en la etapa en que el software se encuentra en desarrollo (Pre-Release) y ya se ha superado la etapa de especificación.

Por lo tanto, se analizará en forma sintética cada modelo de proceso para ver como resuelven un cambio en los requerimientos en esta fase del plan de proyecto. Se recuerda aquí que los modelos de procesos son teóricos y que no se suelen utilizar tal cual los describe la teoría sino que se utilizan versiones ad-hoc de éstos que son ajustados por cada organización de desarrollo de software. La utilidad entonces de este análisis es poner en relieve cuales se cree que son las mejores prácticas para el abordaje del problema de la volatilidad de los requerimientos.

Modelo en Cascada

Ante cambios en los requerimientos cuando el proceso ya ha superado la etapa de especificación, este modelo requiere realizar un retroceso a la mencionada fase de especificación de requerimientos, es

decir requiere se detecten las nuevas necesidades, documenten, diseñen y re-planifiquen las tareas de desarrollo.

Tal situación provoca que el proyecto tenga un impacto muy grande en cuanto a costos y no se tenga una real certeza de los tiempos de finalización del mismo, debido a la incertidumbre acerca de nuevos cambios en los requerimientos.

Se infiere entonces que el modelo en cascada es una metodología que no contempla ningún tipo de práctica para la gestión de cambios en los requerimientos.

Modelo Incremental

El modelo incremental comienza con una especificación global de los requerimientos de modo de "plantar" la arquitectura del software. Luego, se priorizan las partes del software y en cada incremento se desarrolla cada parte completamente.

Esta práctica, permite que la definición completa se realice tardíamente dando tiempo a los cambios en los negocios. A esto se le suma que entre el momento de la especificación y la implementación hay un lapso corto de tiempo con lo cual es menos frecuente un cambio entre la especificación y el desarrollo.

De esta manera, los cambios en los requerimientos que se presenten en un proyecto que utiliza para su gestión un modelo incremental por la naturaleza evolutiva del proceso y la constante participación del usuario del producto de software para evaluar cada una de las entregas resultantes de cada iteración son detectados en forma casi inmediata.

Ante estos cambios en los requerimientos se genera la documentación requerida, debido a que esta metodología toma el método de control del modelo en cascada, y se añade un nuevo ciclo al proceso donde el cambio introducido será evaluado por el usuario que lo ha solicitado.

Vale destacar que las soluciones adoptadas son realizadas mediante un fuerte análisis de riesgos entre las alternativas posibles, con lo cual se gana en fiabilidad y calidad de producto final.

Pero como es sabido, la implementación de partes del software tempranamente, aumenta la necesidad de los usuarios por mejorar ese software, ya sea agregando funciones o modificando las existentes incrementando exponencialmente la volatilidad.

Metodologías agiles

Las metodologías agiles son por naturaleza procesos incrementales, entonces aborda el problema de la volatilidad de requerimientos con los mismos principios, es decir manejar el proceso de desarrollo vía entregas incrementales y con una activa participación del usuario.

La particularidad que distingue a este tipo de modelo de desarrollo es que dejan de lado los aspectos formales que tienen que ver con la documentación y planificación que en cierta medida tiene probabilidad de cambiar durante el proceso total.

Otro aspecto importante en que se basa este tipo de metodología es que propone que con cada iteración se deba realizar una evaluación acerca de la arquitectura del software, a los efectos de determinar si la misma se verá afectada en cuanto a estructura por los cambios que deban introducirse y de ser necesario realizar una refactorización de la arquitectura. Con esta práctica se obtiene que el producto final tenga la calidad esperada y disminuye la probabilidad de aparición de errores luego de la liberación del producto de software.

Por lo mencionado en el precedente análisis se puede concluir que los modelos incrementales cuentan con las mejores prácticas para el abordaje del problema de la volatilidad de los requerimientos.

- Activa participación del usuario en el proceso de desarrollo.
- Detección temprana de cambios en las necesidades del usuario.
- Incorporación de tales cambios en una iteración subsiguiente.
- Análisis de impacto de las modificaciones a introducir.

Aunque el costo y tiempo planificado al inicio del proyecto se vea afectado, esto se ve soslayado por el valor agregado que brinda la posibilidad de que el cliente pueda tener disponible partes importantes del software tempranamente.

Dependiendo de las necesidades de la organización donde se lleve a cabo el desarrollo se deberá optar por el tipo de proceso incremental a utilizar. Es decir para una organización donde se requiera la documentación detallada del desarrollo del producto de software se recomienda un modelo en espiral, y para organizaciones donde la mencionada documentación no sea requisito se puede optar por una metodología ágil.

Como trabajo futuro se plantea la realización de experiencias de laboratorio para obtener métricas en cuanto a cómo afectan a los proyectos los cambios que puedan surgir en los requerimientos, utilizando la estrategia propuesta de modo de verificar si la volatilidad en los requerimientos fue reducida en alguna medida o en su defecto si los cambios en los requerimientos fueron gestionados de una forma que no signifiquen una amenaza para el éxito del proyecto.

AGRADECIMIENTOS

A mi esposa e hijas, por la paciencia y el apoyo.

A mis padres por inculcarme la cultura del trabajo y estudio.

A mis profesores por la calidad de formación.

A mi director por su tiempo, colaboración y guía.

REFERENCIAS

Barry, E. J. (2002); Slaughter, A.,"Software Project Duration and Effort: An Empirical study", Information Technology and Management, vol. 3, 1-2, pp. 113-136.

Fagalde, P. (2011); "Artefactos de especificación de requerimientos de usabilidad", Universidad de Buenos Aires – Facultad de Ingeniería, Tesis de Ingeniería en Informática, Bs As.

Hammer, T. (1998); Huffman, L.; Rosenberg, L., "Doing Requirements Right the First Time", CROSSTALK, The Journal of Defence Software Engineering.

Herrera, L. J. (2003),"Ingeniería De Requerimientos - Ingeniería De Software". http://www.ilustrados.com/tema/1605/Ingenieria-Requerimientos-Ingenieria-Software.html (Octubre, 2013)

Ali Javeed, M. (2006); "Metrics for Requirements Engineering", Department of Computing Science Umeå University SE-90187 Umea, Sweden.

Javed, T. (2004); Manzil, M.; Durrani, S.,"A Study to Investigate the Impact of Requirements Instability on Software Defects", National University of Computer and Emerging Sciences, 852-B, Pakistan

Mizuno, Y. (1983), "Software Quality Improvement", IEEE Computer, Vol. 16, No. 3, pp 66-72.

Mundlamuri, S. (2005), "Managing the Impact of Requirements Volatility", Department of Computing Science Umeå University SE-90187 Umea, Sweden.

Nidumolu, S. (1996), "Standardization, Requirements Uncertainty and Software Project Performance," Information & Management, vol. 31, pp. 135-150.

Nurmuliani, N. (2007); Zowghi, D; Fowell, S., "Analysis of Requirements Volatility during Software Development Life Cycle", Faculty of Information Technology University of Technology, Sydney. http://www.robertfeldt.net/courses/reqeng/papers/nurmulian_2004_analysis_of_req_volatility.pdf (Noviembre 2013).

Pressman, R.S. (2006), "Ingeniería de Software. Un enfoque práctico", Editorial McGraw-Hill, Madrid.

Rajlich, V. T. (2006), "Changing the paradigm of software engineering", Communications of the ACM, August 2006/Vol. 49, No. 8.

Sawyer P. y Kontoya G. (1999), "SWEBOK: Software Requirements Engineering Knowledge Area Description", Computing Department Lancaster University, Version 0.5.

Singh, M. P. (2012); Vyas M., “Requirements Volatility in Software Development Process”, International Journal of Soft Computing and Engineering, Volume-2, Issue-4, September 2012.

Sommerville, I. (2011), “Ingeniería del Software”, Editorial Pearson, Boston.

Stark, G. (1999); Oman, P.; Skillicorn, A.; Ameele, A., “An Examination of the Effects of Requirements Changes on Software Maintenance Releases” Journal of Software Maintenance Research and Practice, Vol. 11, 1999, pp: 293-309.

TSG (1998), "CHAOS: A Recipe for Success", The Standish Group.

Zowghi, D. (2007); Nurmuliani, N., “A Study of the Impact of Requirements Volatility on Software Project Performance”, Faculty of Information Technology, University of Technology, Sydney.