VENTAJAS Y DESVENTAJAS DE CADA UNO DE LOS CICLOS DE VIDA. CRITERIOS PARA ELECCIÓN DE CICLOS DE VIDA EN FUNCIÓN DE LAS NECESIDADES DEL PROYECTO Y LAS CARACTERÍSTICAS DEL PRODUCTO.

Luciana Agüero – 67353
luuchita1995@gmail.com
Gastón Cabrera – 67051
gasty065@gmail.com
Genaro Franceschelli – 67099
gena.franceschelli@gmail.com
Maria Lucia Okamoto – 66884
luuokamoto95@gmail.com
Rodrigo Miguel Vega Gimenez – 67027
rvega389@gmail.com
Maximiliano Yacuzzi – 66276
maxiyacuzzi15@gmail.com

RESUMEN: *En el presente paper describimos cada uno de los ciclos de vida o modelos de proceso e identificamos sus ventajas y desventajas dentro del marco del proceso de desarrollo de software. A su vez hacemos una breve introducción al tema explicando algunos conceptos básicos que nos ayudarán a definir los criterios para la elección de un determinado ciclo de vida de acuerdo a las necesidades del proyecto y las características del producto.*

PALABRAS CLAVE: Ciclo de vida, proceso, producto, proyecto.

1 INTRODUCCIÓN

Durante finales de la década de los 60, se pudo visualizar en el área computacional un avance considerable en cuanto al potencial del hardware, generando la posibilidad de construir software más complejo y de mayor tamaño, que no vino acompañado de una mejora significativa ni mucho menos equiparable en el desarrollo del software. Esto causó que el software que se desarrollaba sea de mala calidad, y finalmente olvidado. F.L. Bauer recalcó la dificultad de generar software libre de defectos, fácilmente comprensibles y que sean verificables. La falla enorme existente en ese momento era la concepción que se tenía del proceso de desarrollo de software, el cual era deficiente e incluso inexistente en muchos casos.

Si se puede resumir de alguna manera, solo el 25% del tiempo de desarrollo era destinado a las fases de análisis, diseño, implementación y pruebas, mientras que el restante 75% era dedicado a correcciones y mantenimiento. Esto provocaba constantes paradas y retrocesos para revisar los errores graves que eran acarreados desde las primeras etapas del "proceso". Ante este panorama, fue necesaria la introducción de nuevas herramientas que solucionen de alguna manera la caída libre que estaba sufriendo el desarrollo de software.

La introducción de nuevas dinámicas al proceso de desarrollo de software vino aparejada con la imperiosa necesidad de generar productos de calidad, es decir, aquel que cumple los requisitos funcionales y de rendimiento que fueron establecidos previamente. La incorporación al proceso de nuevos modelos de desarrollo y modificación del ciclo de vida, entre otros avances, permitieron que el desarrollo de software sea mucho mas metodológico y estructurado, disminuyendo de forma notable los fallos y correcciones con un costo elevado. [1]

A continuación describiremos el concepto de proceso, proyecto y producto y mencionaremos algunos aspectos importantes de dichos conceptos, para ello nos basaremos principalmente en la bibliografía sugerida por la cátedra de Ingeniería de Software de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Córdoba.

La redacción del presente paper se hará de acuerdo al estándar IEEE.

Se intentará además desarrollar los modelos de procesos que se pueden aplicar a los proyectos de desarrollo de software de manera genérica con sus ventajas y desventajas, para luego poder adaptarlos al contexto de trabajo. Daremos también una opinión personal y una apreciación acerca de los criterios a aplicar para la elección del ciclo de vida más conveniente según las necesidades del proyecto y las características del producto.

2 PROYECTO

Un proyecto es un conjunto de tareas interrelacionadas basadas en esfuerzos, limitado en el tiempo, con un objetivo definido, que requiere del acuerdo de un conjunto de especialidades y recursos. También puede definirse como una organización temporal con el fin de lograr un propósito específico. Cuando los objetivos de un proyecto son alcanzados se entiende que el proyecto está completo. Los objetivos guían al proyecto. Un proyecto se lleva a cabo para crear un producto, servicio o resultado único.

Los proyectos de software pueden ser de diferentes tipos, grado de dificultad, urgencia, prioridad y tamaños, sin importar estos atributos los mismos suelen estar conducidos por las fechas fijas impuestas y los periodos de cambios frecuentes.

3 PRODUCTO

El término software no solo hace referencia a programas de computadora, sino que además se refiere a todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general, un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes.

Los ingenieros de software se concentran en el desarrollo de productos de software, es decir software que se vende a un cliente. [2]

4 PROCESO

Un proceso es una secuencia de pasos ejecutados para un propósito dado, por lo que un proceso de software se define como un conjunto de actividades, métodos, prácticas y transformaciones que se utilizan para desarrollar y mantener software.

Estas actividades varían dependiendo de la organización y el tipo de sistema que debe desarrollarse. Sin embargo existen cuatro actividades fundamentales de proceso que son comunes para todos los procesos del software. Estas actividades son:

- Especificación de software donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
- Desarrollo del software donde el software se diseña y programa.
- Validación del software donde el software se valida para asegurar que es lo que el cliente requiere.
- Evolución del software donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado. [3]

5 RELACIÓN PROYECTO – PRODUCTO – PROCESO

Un proyecto de software se adapta a un proceso de desarrollo de software y del mismo se obtiene como resultado un producto.

Como se expresó anteriormente en este informe un proyecto es un conjunto de tareas relacionadas y ejecutadas para cumplir con un objetivo, ya sea este un producto o servicio. Un proyecto es siempre único y temporal dado que tiene un comienzo y un fin definido, y por lo tanto tiene un alcance y recursos determinados.

Asimismo, cuando hablamos de proceso, nos referimos al conjunto de actividades relacionadas y rutinarias, por las cuales se pueden realizar varios proyectos similares. Se trata de una metodología o procedimiento a seguir para obtener un tipo de producto o servicio. La rutina de trabajo que engloba todos los recursos y requerimientos para cumplir con sus objetivos.

6 MODELO DE PROCESO DE SOFTWARE

Un modelo de proceso de software es una descripción simplificada de un proceso de software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería de software. [4]

Estos modelos generales no son descripciones definitivas de los procesos del software. Más bien, son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo de software. Puede pensarse en ellos como marcos de

trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería de software. [5]

Los modelos de proceso también son conocidos como modelos de ciclo de vida ya que constituyen una serie de pasos a través de los cuales el producto o proyecto progresa. Los modelos especifican las fases de proceso y el orden en el que se llevan a cabo.

7 CLASIFICACIÓN DE LOS CICLOS DE VIDA

Los ciclos de vida que se presentan a continuación no son mutuamente excluyentes y con frecuencia se usan en conjunto, sobre todo para sistemas complejos y de gran envergadura.

7.1 CICLO DE VIDA SECUENCIAL

Toma las actividades fundamentales del proceso; especificación, desarrollo, validación y evolución y los representa como fases separadas del proceso: especificación de requerimientos, diseño de software, implementación, pruebas, etc.

Ventajas:
- Desarrollo dirigido por un plan, por eso es denominado secuencial.
- De cada etapa vamos a obtener documentación para realizar un monitoreo constante contra el plan.
- Muy útil cuando los requerimientos son claros y es poco probable un cambio drástico durante el desarrollo.

Desventajas:
- La documentación puede ser burocrática y excesiva, esto hace que la mayoría tienda a menospreciar la necesidad de mantener una buena documentación, pero el equilibrio se encuentra en hacer que no sea excesiva.
- En etapas finales puede ser que haya que rehacer trabajo por cambios en requerimientos o fallas de diseño. Realizar esta retrospección hacia las primeras etapas puede llegar a ser muy costoso.

7.2 CICLO DE VIDA ITERATIVO / INCREMENTAL

Este enfoque vincula las actividades de especificación, desarrollo y validación. El sistema se desarrolla como una serie de versiones (incrementos) y cada una añade funcionalidades a la versión anterior.

Ventajas:
- La especificación, desarrollo y validación están entrelazadas en lugar de separadas y aisladas.
- Rápida retroalimentación a través de las actividades.
- Muy útiles para sistemas de requerimientos cambiantes o cuando existe incertidumbre con respecto a ellos.
- Más fácil y menos costoso implementar cambios.
- Cada iteración genera funcionalidad para el cliente.

Desventaja:
- Los incrementos progresivos tienden a degradar la estructura del sistema.

7.3 CICLO DE VIDA RECURSIVO

Se inicia con algo en forma completa, como una subrutina que se llama a si misma e inicia nuevamente. Se presenta un prototipo que va mejorando con cada vuelta.

Ventajas:
- Se generan productos independientes de la implementación, que pueden ser reusables en sistemas de características similares.

Desventajas:
- Puede ser más costoso en tiempo y dinero readaptarlos para reutilizarlos para los diferentes proyectos.
- La tecnología puede ser obsoleta.
- Pueden carecer de mantenimiento o documentación.

8 MODELOS DE CICLOS DE VIDA

Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso.

En otras palabras todos los modelos del proceso del software pueden incluir las actividades estructurales generales pero cada una pone distinto énfasis en ellas y define en forma diferente el flujo de proceso que invoca cada actividad estructural (así como acciones y tareas de ingeniería de software). [6]

En los apartados subsiguientes se desarrollarán los modelos más relevantes descriptos por Roger S. Pressman en el libro titulado “Ingeniería del software, un enfoque práctico”, 7ma edición.

8.1 MODELO EN CASCADA

El modelo de la cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del

cliente y avanza a través de planeación modelado, construcción y despliegue, para concluir con el apoyo del software terminado. [7]

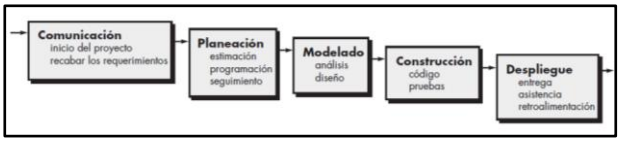

Figura 1. Modelo en cascada.

Entre las ventajas que podemos encontrar de este modelo se puede mencionar que la documentación se produce como resultado en cada fase y que su uso es recomendable cuando los requerimientos se puedan comprender correctamente y sea casi improbable que los mismos cambien radicalmente durante el desarrollo del sistema.

Como desventajas del modelo podemos mencionar que no es posible empezar con la siguiente fase hasta que la fase previa no hay finalizado esto puede llegar a provocar "estados de bloqueo" en los que ciertos miembros del equipo de proyecto deben esperar a otros a fin de terminar tareas interdependientes.

Además debido a los costos de producción y aprobación de documentos, las iteraciones son costosas e implican rehacer el trabajo. Por lo tanto, después de un número reducido de iteraciones, es normal congelar partes del desarrollo, como la especificación, y continuar con las siguientes etapas de desarrollo. Los problemas se posponen para su resolución, se pasan por alto o se programan.

Por otra parte se sabe que es difícil para el cliente enunciar en forma explícita todos los requerimientos y el modelo en cascada necesita que esto se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos. Es esencial que en este modelo el cliente tenga paciencia ya que no se dispondrá de una versión funcional del (de los) programa (s) hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento.

Una variante de la representación del modelo en cascada es el modelo en V. En la figura 2 se ilustra el modelo en V donde se aprecia la relación entre las acciones para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana. A medida que el equipo de software avanza hacia abajo desde el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema y de su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, y en esencia ejecuta una serie de pruebas (acciones para asegurar la calidad) que validan cada uno de los modelos creados cuando el equipo fue hacia abajo por el lado izquierdo. [8]

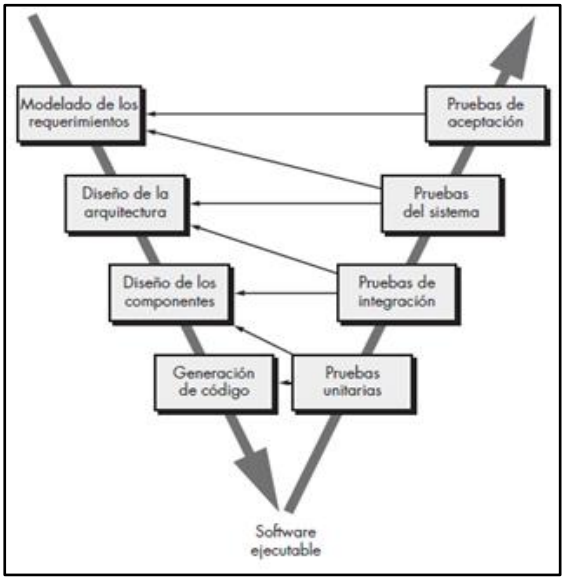

Figura 2. El modelo de proceso en V.

No hay demasiadas diferencias entre el ciclo de vida clásico y el modelo en V. Las ventajas y desventajas son las mismas para ambos modelos.

8.2 MODELO DE PROCESO EVOLUTIVO

El software, como todos los sistemas complejos, evoluciona en el tiempo. Es frecuente que los requerimientos del negocio y del producto cambien conforme avanza el desarrollo, lo que hace que no sea realista trazar una trayectoria rectilínea hacia el producto final; los plazos apretados del mercado hacen que sea imposible la terminación de un software perfecto, pero debe lanzarse una versión limitada a fin de aliviar la presión de la competencia o del negocio; se comprende bien el conjunto de requerimientos o el producto básico, pero los detalles del producto o extensiones del sistema aún están por definirse. En estas situaciones y otras parecidas se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo.

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software. [9] Se presentaran dos modelos comunes de proceso evolutivo.

8.2.1 PROTOTIPOS

El ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software. Si va a construirse un prototipo, pueden utilizarse fragmentos de

programas existentes o aplicar herramientas (por ejemplo, generadores de reportes y administradores de ventanas) que permitan generar rápidamente programas que funcionen.

El prototipo sirve como "el primer sistema". Lo que Brooks recomienda es desecharlo. Pero esto quizá sea un punto de vista idealizado. Aunque algunos prototipos se construyen para ser "desechables", otros son evolutivos; es decir, poco a poco se transforman en el sistema real.

Tanto a los participantes como a los ingenieros de software les gusta el paradigma de hacer prototipos. Los usuarios adquieren la sensación del sistema real, y los desarrolladores logran construir algo de inmediato. [10]

8.2.2 EL MODELO ESPIRAL

El modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas.

Con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo.

El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que el proceso avanza, el desarrollador y cliente comprenden y reaccionan mejor ante los riesgos en cada nivel de evolución. El modelo espiral usa los prototipos como mecanismo de reducción de riesgos, pero, más importante, permite aplicar el enfoque de hacer prototipos en cualquier etapa de la evolución del producto. El modelo espiral demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y, si se aplica de manera apropiada, debe reducir los riesgos antes de que se vuelvan un problema. [11]

Continuando con los modelos de procesos evolutivos podemos afirmar que los mismos brindan la posibilidad de que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se puede reflejar en el sistema software. Por otra parte, su uso es recomendable en sistemas pequeños y de tamaño medio (500.000 líneas de código).

Además es posible aseverar que el enfoque evolutivo de desarrollo no es viable en sistemas grandes y complejos con un periodo de vida largo, donde diferentes equipos desarrollan distintas partes del sistema. Es difícil establecer una arquitectura del sistema estable, lo cual hace difícil integrar las contribuciones de los equipos.

8.3 MODELO CONCURRENTE

El modelo de desarrollo concurrente, en ocasiones llamado ingeniería concurrente, permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso descritos anteriormente.

La figura 3 muestra la representación esquemática de una actividad de ingeniería de software dentro de la "actividad de modelado" con el uso del enfoque de modelado concurrente. La actividad (modelado) puede estar en cualquiera de los estados mencionados en un momento dado. En forma similar, es posible representar de manera análoga otras actividades, acciones o tareas (por ejemplo, comunicación o construcción). Todas las actividades de ingeniería de software existen de manera concurrente, pero se hallan en diferentes estados.

El modelado concurrente define una serie de eventos que desencadenan transiciones de un estado a otro para cada una de las actividades, acciones o tareas de la ingeniería de software.

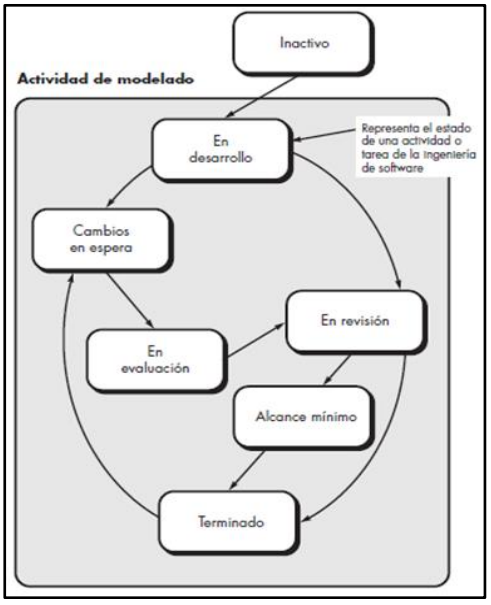

Figura 3. El modelo de desarrollo concurrente.

El modelado concurrente es aplicable a todos los tipos de desarrollo de software y proporciona un panorama apropiado del estado actual del proyecto. En lugar de confinar las actividades, acciones y tareas de la ingeniería de software a una secuencia de eventos, define una red del proceso. Cada actividad, acción o

tarea de la red existe simultáneamente con otras actividades, acciones o tareas. Los eventos generados en cierto punto de la red del proceso desencadenan transiciones entre los estados. [12]

Los modelos concurrentes son excelentes para proyectos en los que se conforman grupos de trabajo independientes y proporcionan una imagen exacta del estado actual de un proyecto. En esencial de que existan grupos de trabajo ya que de lo contrario no se podría trabajar en este modelo.

8.4 MODELO BASADO EN COMPONENTES

El modelo de desarrollo basado en componentes incorpora muchas de las características del modelo espiral. Es de naturaleza evolutiva y demanda un enfoque iterativo para la creación de software. Sin embargo, construye aplicaciones a partir de fragmentos de software prefabricados. [13]

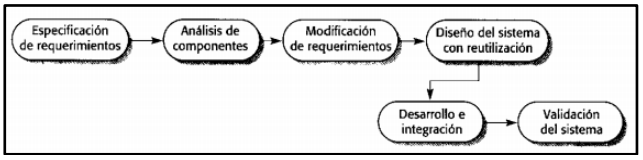

Figura 4. Modelo basado en componentes.

El modelo basado en componentes lleva a la reutilización del software, y eso da a los ingenieros de software varios beneficios en cuanto a la mensurabilidad. Gracias a la reutilización, el equipo de ingeniería de software tiene la posibilidad tanto de reducir el ciclo de tiempo del desarrollo como el costo del proyecto. Por ende permite una entrega más rápida del software, reduce la cantidad de software a desarrollarse y por lo tanto se reducen los costos y los riesgos.

Sin embargo si las nuevas versiones de los componentes reutilizables no están bajo el control de la organización que los utiliza, se pierde parte del control sobre la evolución del sistema.

9 ELECCIÓN DE CICLOS DE VIDA

La ingeniería de software es una disciplina de la ingeniería que se preocupa de todos los aspectos de la producción de un software; desde las primeras etapas de la especificación hasta el mantenimiento del sistema después que se pone en operación.

En otras palabras, es la aplicación de enfoques sistemáticos y disciplinados al desarrollo de software, para esto se han creado modelos y metodologías para la correcta utilización del tiempo y recursos que una empresa o entidad disponen.

Los modelos de desarrollo de software ofrecen un marco de trabajo usado para controlar el proceso de desarrollo de sistemas de información, estos marcos de trabajo consisten en una filosofía de desarrollo de programas.

La elección del ciclo de vida depende de la estrategia y este debe permitir seleccionar los artefactos a producir, definir actividades y roles, y modelar conceptos.

Hay distintos modelos de ciclo de vida que son utilizados para el desarrollo de un sistema software, estos definen el orden de las tareas o actividades involucradas, la coordinación entre ellas, su enlace y realimentación. [14] Algunos de los factores que influyen a la hora de elegir un ciclo de vida son:

- Disponibilidad de recursos ya sean económicos (presupuesto disponible), tiempo, equipos, humano, etc.
- Conocer y entender los requerimientos.
- Dominio del problema.
- Complejidad y magnitud del proyecto.
- Necesidad de documentación.
- Adaptabilidad a los cambios.
- Ciclo de tiempo requerido.
- Aspectos del cliente.
- Riesgos, ya sean técnicos o de administración.

Cada quien es libre de utilizar cualquier modelo de desarrollo de software ya que lo que se busca es optimizar el proceso de desarrollo conforme a los requerimientos establecidos, logrando así desarrollar un sistema de mayor calidad. [15] Un proyecto de alta calidad es aquel que entrega el producto requerido, el servicio o resultado, satisfaciendo los objetivos en el tiempo estipulado y con el presupuesto planificado.

10 CONCLUSIÓN

A partir de la consideración del software como producto final de un proyecto surge la necesidad de implementar metodologías y técnicas de desarrollo que nos aseguren que los recursos disponibles serán utilizados de la forma más eficiente y eficaz posible con el fin de obtener resultados satisfactorios. La selección apropiada de una determinada metodología es un proceso complejo, puesto que los proyectos de software son de múltiples naturalezas, tamaños y requerimientos.

Definir el proceso y el ciclo de vida es una etapa fundamental en la planificación de un proyecto, el mismo define que trabajo técnico debería realizarse en cada fase, quien debería estar involucrado en cada fase, como controlar y aprobar cada fase, como deben generarse los entregables y como revisar, verificar y validar el producto final obtenido.

Es importante tener en cuenta que la selección de una metodología es un proceso que se debe llevar a cabo por personal capacitado, puesto que no todas ofrecen los mismos resultados bajo idénticas condiciones y en cualquier proyecto se cuenta con recursos limitados cuyo aprovechamiento resulta conveniente maximizar. Por otra parte no es posible generalizar una misma solución a un conjunto de proyectos que a simple vista parecieran ser similares, ya que llevar a cabo esto implica que suponemos que los proyectos que queremos solucionar son resueltos de la misma manera, cuando en realidad cada proyecto es único y por lo tanto se deberán implementar resoluciones diferentes para cada uno de ellos.

11 REFERENCIAS

[1] Crisis del Software. [En línea] Disponible en: https://histinf.blogs.upv.es/2011/01/04/la-crisis-del-software/
[2] Sommerville, Ian. *"Ingeniería de Software"*. 7ma Edición. Editorial Pearson Addison Wesley. pp 5-6.
[3] Sommerville, Ian. *"Ingeniería de Software"*. 7ma Edición. Editorial Pearson Addison Wesley. pp 7.
[4] Sommerville, Ian. *"Ingeniería de Software"*. 7ma Edición. Editorial Pearson Addison Wesley. pp 8.
[5] Sommerville, Ian. *"Ingeniería de Software"*. 7ma Edición. Editorial Pearson Addison Wesley. pp 61.
[6] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 33.
[7] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 34.
[8] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 34.
[9] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 36.
[10] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 37-38.
[11] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 39-40.
[12] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 41-42.
[13] Roger S. Pressman. *"Ingeniería del software, un enfoque práctico"*. 7ma edición. Ed. McGRAW-HILL Interamericana Editores, S.A. pp 43.
[14] Criterios para la selección de un modelo de ciclo de vida de desarrollo de software. [En línea] Disponible en: http://zonalista.blogspot.com/2012/10/criterios-para-la-seleccion-de-un.html
[15] Criterios para la selección de un modelo de ciclo de vida de desarrollo de software. [En línea] Disponible en: http://zonalista.blogspot.com/2012/10/criterios-para-la-seleccion-de-un.html