TESIS DE GRADO

PROYECTO:

MODELO DE GESTIÓN PARA EMPRESAS DE SOFTWARE EN EL ECUADOR

Andrés Bastidas F.

VI MAESTRIA EN ALTA GERENCIA

MAYO 2011

Autoría

El presente proyecto fue desarrollado por el Ing. Andrés Fernando Bastidas Fuertes, basado en el planteamiento investigativo y contrastado con su amplia experiencia profesional.

Su rama profesional es de Ingeniero en Sistemas y Computación, contando con 6 años de experiencia en la gerencia de una empresa de software en el Ecuador, y la participación de decenas de proyectos de software de todo tipo y para múltiples industrias, cumpliendo roles desde desarrollador de sistemas hasta gerente de proyecto.

Dedicatoria

Dedico mi tesis de grado de Maestría al grupo de trabajadores en el área de tecnología y desarrollo de software, que realizan día a día grandes esfuerzos para que los productos tecnológicos ecuatorianos sean reconocidos internacionalmente, inclusive trabajando en muchas ocasiones hasta altas horas de la noche para cumplir con este objetivo.

En especial dedico este trabajo a mi equipo de trabajo, a todos los profesionales que han pasado por la empresa SmartWork S.A., que con su destacada dedicación y alto compromiso fue posible obtener productos de calidad internacional, enfatizando que gracias a ellos me fue posible obtener una amplia experiencia en la gerencia de proyectos de software, que ahora he tratado de plasmar en este documento.

Agradecimiento

Agradezco a mi esposa Andrea, a mi hija Samantha, a mis padres, a mis hermanos, a familiares naturales y políticos, y amigos, por su constante apoyo en todas las aventuras y episodios de mi vida, en especial ahora en la culminación de éste reto profesional, para el cual tuve un apoyo incondicional, y que significa para mí un gran paso en mi proyecto de vida.

Resumen Ejecutivo

La producción y comercialización de software tiene características particulares que han obligado a los gerentes de las empresas desarrolladoras a diseñar modelos específicos para gestionar este tipo de empresas. Estas distinciones en la gestión empresarial son necesarias por tratarse de un producto que tiene un diseño en constante variación y un nivel de tecnicismo muy profundo, ocasionando retrasos y desfases presupuestarios.

La mayoría de estos desfases son consecuencia de las decisiones gerenciales tomadas en dos etapas específicas: en la Etapa de definición del proyecto, específicamente en el análisis para la distribución de tiempos y estructura presupuestaria; y en la Etapa de ejecución, particularmente en la relación interna de los equipos de trabajo con la gestión y aplicación del conocimiento.

El objetivo de esta investigación consiste en determinar los métodos gerenciales utilizados en el Ecuador, con la finalidad de hallar soluciones factibles y consolidación de experiencias.

Luego de realizar una investigación por medio de una encuesta, se encontró que los atrasos en los proyectos de software son reales, recurrentes y en algunos casos alarmantes, encontrando la existencia de una deficiencia teórica / práctica que permita solventar el problema fiablemente.

Es por ello que se unió la experiencia y la información estadística para establecer 16 lecciones que deben ser consideradas como “buenas prácticas gerenciales” para este tipo de empresas.

Palabras clave

- Modelo gestión
- Desarrollo de software
- Desfases de tiempo y presupuesto
- Medición de proyectos de Software
- Buenas prácticas gerenciales

Índice

Epígrafes

"Se dice que el gestionar una empresa muchas veces depende más del sentido común que de cualquier otra habilidad gerencial. Mi opinión es que uno no nace con un 'buen sentido común', hay que estudiar mucho y experimentar más para tenerlo":

Andrés Bastidas F.

PARTE I: Introducción y fundamentación teórica

Capítulo 1. Introducción

Desde el punto de vista contable el software es registrado como un activo intangible de larga duración, pero a diferencia de otros bienes intangibles, es posible tocarlo, verlo, manipularlo, manejarlo y trasladarlo, por lo que se dice entre las personas involucradas en esta industria, que el software se encuentra en la frontera entre lo tangible y lo intangible, ya que comparte características de ambos mundos.

Por ende la producción y comercialización de software, a diferencia de cualquier otro producto o servicio producido, tiene características particulares que han llevado a los gerentes de las empresas productoras a diseñar modelos específicos para gerenciar este tipo de negocios. Estas distinciones en la gestión empresarial son esenciales para lograr una buena rentabilidad, debido a que se trata de un producto que tiene un diseño en constante variación y un nivel de tecnicismo muy profundo, involucrando una gran cantidad de variables que no son vistas en otros escenarios profesionales.

Esto ha obligado a las empresas que trabajan con este tipo de productos a adoptar formas especializadas para casi todas las áreas de gestión: Recursos Humanos, Control sobre la Producción, Comercialización, Planificación y Presupuestación, Contabilización y Gerencia).

Con el fin de ejemplificar la complejidad que conlleva el desarrollo de un paquete de software, se formula como ejemplo comparativo con el proceso

industrial de fabricación de una silla: En primer lugar se reúne un conjunto de diseñadores e ingenieros a decidir las formas, diseños, estándares y materiales de la silla, por lo tanto luego de haberse terminado la etapa de diseño, es posible conocer exactamente cuántas piezas tendrá, qué tipo de materiales utilizará, qué colores y formas presentará, qué maquinarias se utilizarán en su producción, qué funcionalidades y limitaciones prestará, e inclusive a qué precio se venderá y qué utilidad se esperará por cada unidad vendida. Una vez que inicia la línea de producción, la empresa deberá preocuparse por la provisión de materia prima, mano de obra y otros insumos, para que al final el equipo de comercialización coloque los productos en el mercado y lleguen al consumidor final.

Cuando se trata de la 'fabricación de software', se cumple un proceso muy diferente, con las siguientes etapas especializadas:

- En la etapa de visionamiento se describen los grandes lineamientos del software, los requerimientos de los usuarios, los riesgos inherentes al proyecto, recursos disponibles y los factores determinantes de cumplimiento de los objetivos. En esta etapa también se analiza las posibles futuras versiones del software, los modelos de comercialización[1] y las formas como el software deberá ser compatible a futuro.
- En la etapa de diseño se elaboran modelos que permiten bosquejar la estructura del software y la forma como se llegará a una solución

[1] A diferencia de un producto físico, el software tiene varias otras formas de comercializarse en el mercado además de la compra y venta comunes. Ejemplos serían el arrendamiento, comisión por uso de terceros, cobro por transacción, intercambio de espacio, gratuidad en el producto y cobro en el soporte, modelo basado en negocios alternativos de intercambio, modelo de base instalada, entre otros.

tecnológica. En condiciones normales, se torna muy complicado lograr la elaboración de un documento de especificaciones funcionales del software que describa exactamente el producto final esperado[2], esto ocurre porque el nivel de tecnicismo y complejidad implican comúnmente en irse adaptando a cambios durante la ejecución del proyecto. Es por ello que los documentos de especificaciones funcionales constituyen tan solo una guía que orientará qué tipo de herramientas debe contar el sistema y qué camino se tomará para aproximarse al resultado esperado.

- En la etapa de programación, se interpretarán los diseños hechos en la etapa anterior, traduciéndolos a código fuente de programación y en varios puntos es posible que se "modifique el diseño"[3] en tanto sea necesario ajustarlo para llegar a la funcionalidad esperada y al cumplimiento del objetivo. Esta etapa es la fabricación del software en sí, siendo el período donde se involucrará la mayor cantidad de esfuerzo de los programadores en la tarea de representar los modelos diseñados en lenguaje de programación. El resultado es el "código fuente matriz", que funcionará como plantilla para ser compilado a "software ejecutable" y trasladado para cada uno de los clientes que lo puedan correr en sus computadoras.

[2] Cuando se trata de una empresa que desarrolla "software a medida para terceros", este problema se incrementa en mayor medida, en el sentido que al inicio los clientes solicitan la elaboración de cotización, previo a realizar cualquier tipo de estudio. Por ende la mayoría de profesionales que hacen la valoración de este tipo de proyectos se basan en su experiencia o en aproximaciones cuantitativas que involucran la incorporación de márgenes de riesgo intuitivos y valoración conforme a la capacidad del equipo de desarrollo, más no en una teoría formal.

[3] El concepto de que el software no se trata de un producto, sino de un "modelo en constante evolución" lo he determinado en base a mi experiencia obtenida luego de haber trabajado con proyectos de software por más de 10 años. Este es uno de los puntos que se tratará de contrastar durante la presente investigación.

- En la etapa de pruebas se validará que las herramientas programadas se ajusten a los requerimientos deseados y se asegure los procesos funcionen tal como se concibieron. En el ejemplo propuesto de la fabricación de la silla, las pruebas del producto consistirán en el control de calidad en la elaboración del producto al final de la línea de fábrica, de tal manera que valide el cumplimiento del producto con sus especificaciones técnicas. Si una silla no cumple con esta especificación, se considerará como producto de calidad insuficiente y entrará a un reproceso y a producto desechado. En el caso del software, es posible que en este punto se encuentren fallos importantes en la programación, sin que esto signifique que el proyecto el software no es válido, sino por el contrario, es el mecanismo que permite asegurar que el producto llegue a cumplir con la calidad suficiente para entregarse a los usuarios, es por eso que se dice en el medio que las pruebas del software son muchas veces más intensas que la misma etapa de programación, justamente por el nivel de detalle involucrado en estas tareas.
- Una vez terminado el desarrollo, es posible vender el software a uno (desarrollo a medida) o a miles de clientes (masivo) sin incurrir en costos de producción o mano de obra posterior. Entonces puede decirse que una vez terminado el desarrollo de software, el costo de producción se reduce a los centavos que pueda costar la copia de un CD por cada cliente, sin embargo, el cliente paga un valor alto por los modelos, conceptos y mejoras en la automatización, siendo esta la utilidad de la empresa.

- Luego de esto, se regresa a la etapa de visionamiento, para realizar la siguiente versión del software. Este ciclo es repetido indefinidamente, mientras sea necesario ir perfeccionando y complementando las herramientas.

Por lo expuesto, puede decirse que el gerenciamiento de proyectos de software tiene múltiples particularidades en su planificación y ejecución, que no son similares con otros tipos de administración de proyectos o control de la fabricación de productos. Es indispensable que el gerente dilucide y pronostique gran parte de las siguientes variables antes de iniciar un proyecto de este tipo:[4]

- Alta variabilidad e imprecisión en los requerimientos de los clientes: a pesar de los esfuerzos y metodologías que buscan levantar esta información de la forma más completa posible. Llegar a un alto nivel de detalle en la descripción de requerimientos de los clientes puede implicar un esfuerzo tan alto como hacer el software completo.
- Dificultad de conocer a ciencia cierta todos los elementos que serán incluidos durante la etapa de programación: Los profesionales que tienen la tarea de definir los requerimientos para el software, rara vez tienen claridad en lo que quieren, lo que se desea institucionalmente y lo que necesitan las personas que realmente utilizarán el software. El analista de sistemas necesita intuir las necesidades para hacer sus planteamientos y discriminar los requerimientos verdaderos, de

[4] Las aseveraciones presentadas en el listado son resultado de la experiencia adquirida durante el tiempo de administración proyectos de software, y son ideas que se han tratado de corroborar a lo largo de esta investigación con otros gerentes y también con ciertos autores de diversas áreas.

requerimientos circunstanciales del escenario o los actores[5]. El problema de esta metodología es que todo el estudio se basa en la interpretación del analista, que desde su punto de vista externo, puede equivocarse en sus juicios de apreciación y priorización.

- Dificultad para anticiparse a las modificaciones del diseño que son dadas durante la etapa de programación: Este tipo de modificaciones en muchos casos no son posibles predecirlas. Este es uno de los puntos que más impactan en el desarrollo de un sistema, pues si existen modificaciones importantes en una etapa tardía, el esfuerzo de corregir la programación es muy alto y riesgoso.
- Los programadores (unidades de producción) tienen altas variaciones de productividad en función de los diferentes niveles de acumulación de conocimientos, pero también están fuertemente influidas por las diversas condiciones personales, emocionales y culturales.
- Cuando ocurre un cambio de personal dentro de un equipo de trabajo, se afecta de forma considerable la ejecución del cronograma del proyecto, por la dificultad de preparación a una nueva persona sobre el conocimiento acumulado y su adaptación al "ritmo de equipo"[6].
- Tecnologías constantemente cambiantes, que implican mantenerse en frecuente actualización al respecto de los nuevos métodos, técnicas y prácticas.

[5] Se ha tenido la oportunidad de hacer el análisis para el desarrollo de un sistema a la medida, en donde la persona que entregaba los requerimientos a ser incorporados en el software, respondían más a una justificación propia sobre su accionar dentro de la institución, que a resolver las necesidades de información y automatización. En este sentido, existen muchos tipos de entrevistados, dentro de los que aparecen personas de un amplio espectro, desde las que explican superficialmente las necesidades, hasta las que tratan de obtener un solo botón que diga "hacer mi trabajo de hoy".

[6] El "ritmo de equipo" se refiere a la capacidad sinérgica de producción de un equipo de programadores trabajando juntos, que normalmente es superior a la suma de las capacidades de cada uno de sus integrantes.

- Dificultad de prevenir a tiempo las posibles malas decisiones de diseño; en algunos casos estas decisiones son evidenciables recién cuando el producto ya es distribuido a los clientes finales, o en ciertos casos hasta luego de varios meses de uso de la herramienta.

Para apoyar a los gerentes en la elaboración de una estrategia adecuada ante tal laberinto de condiciones, se han realizado muchos estudios a nivel mundial que han planteado modelos matemáticos de predicción y metodologías completas para la planificación de proyectos de software, especialmente enfocados en responder cuánto tiempo y presupuesto debe considerarse durante la formulación de un proyecto de este tipo, y cómo controlar y corregir las variaciones durante la ejecución.

Luego de considerar la experiencia obtenida en 6 años de gerenciamiento de la empresa ecuatoriana llamada SmartWork S.A., especialista en desarrollo de software a medida, se ha percibido que el uso de los modelos matemáticos planteados internacionalmente muy pocas veces se ajustan a la realidad de los proyectos de software en el Ecuador (además de ser muy complicados de manejar), considerando inclusive que la mayoría de gerentes locales que se ha tenido la oportunidad de conversar de este tema directamente, realiza las estimaciones de sus proyectos en función de su experiencia en el área, en su percepción personal o en modelos propios no basados en un desarrollo teórico formal. Esta práctica aparentemente ha dado mejores resultados en el Ecuador que los modelos matemáticos existentes.[7]

[7] Se ha tenido la oportunidad de conocer hace un tiempo a un gerente de una empresa de desarrollo de software ecuatoriana que utilizaba una metodología de medición de presupuesto, basándose en un método

Una de las consecuencias más preocupantes al respecto de estas posibles fallas de planificación, es que se disminuye del tiempo y presupuesto disponible en plena marcha del proyecto, ocasionando que los equipos de programadores de software tengan jornadas de trabajo de 10 a 14 horas diarias (normalmente sin pago de horas extras) para cumplir con las fechas típicamente denominadas “dead-line”.

No hay que perder de vista que este tipo de proyectos implica un alto nivel de exigencia intelectual, lo que implica la ejecución de jornadas de trabajo intensas y por ende acumulación de cansancio, trayendo como resultado que la productividad de los programadores va disminuyendo mientras transcurre el tiempo del cronograma.

La consecuencia evidente es el sentimiento de ‘retraso’ por parte del coordinador o auspiciante del proyecto, lo que acarrea habitualmente que se tome la decisión de incrementar la presión en el equipo de programadores, para que dispongan de más horas de trabajo diaria para acelerar el cumplimiento de metas, pero consiguiendo también la desarticulación de la vida personal de los empleados, causando en muchos casos un conjunto de problemas de estabilidad emocional y profesional[8]. Este incremento paulatino de las exigencias y horarios por los líderes del proyecto, resultan a la larga contraproducentes porque los programadores reducen más aún su nivel de

matemático planteado por una institución internacional, por lo que me interesé en consultarle la forma como lo llevaba a cabo para proyectos nacionales, encontrando que utilizaba un modelo internacional para llegar a una “primera opinión”, pero de todas maneras ajustaba el resultado final basado en su experiencia y percepción de valor de mercado, a la final terminando muy diferente al resultado del método matemático.

[8] Durante mi experiencia profesional, vi muchos casos de personas que tuvieron problemas intensos de desarticulación personal, que terminaron por retirarse de los proyectos y dedicarse a temas no relacionados con la tecnología, o al menos en dejar de programar como medio de sustento profesional.

producción y más bien se generan en desfases de tiempo y complicaciones en el seguimiento del cronograma, difundiendo el 'descontrol', volviéndose en un círculo vicioso.

Aquí se encuentra uno de los mayores dilemas de la industria referente a los Recursos Humanos que se dedican a la programación de sistemas, que es la contradicción entre incrementar la cantidad de horas de trabajo para cumplir con las fechas "dead-line" y disminuirlas para evitar una sobresaturación de la gente mejorando la concentración.[9]

Los gerentes de este tipo de empresas tienen que jugar con estos problemas todos los días, sin disponer de un margen de acción que les permita equilibrar el demandante tiempo que requiere el desarrollo de software versus la estabilidad emocional y física de los desarrolladores; esto sumado a la inexistencia de una metodología eficaz de planificación que se ajuste a la realidad ecuatoriana, plantean un escenario poco alentador para los profesionales de esta área, y en sí para la teoría gerencial.

Luego de observar personalmente y de manera informal varios casos de desarrollo de software en el país, se ha percibido que la mayoría de los desfases de tiempo y presupuesto en los proyectos de software son consecuencia de las decisiones gerenciales tomadas en dos etapas específicas:

- Etapa de definición del proyecto, específicamente en el análisis para la distribución de tiempos y estructura presupuestaria.

[9] En esto hay una gran diferenciación entre una empresa y otra, pues eso ha dependido del modelo de gestión planteado por cada gerencia.

- Etapa de ejecución, particularmente en la relación interna de los equipos de trabajo con la gerencia del proyecto, el control del esfuerzo y la administración del conocimiento.

Por ello se propone la elaboración de la presente investigación acerca de los métodos utilizados en el Ecuador por varias empresas relacionadas al desarrollo de software, con la finalidad de consolidar y difundir las “buenas prácticas” utilizadas en el sector, y encontrar recetas prácticas que ayuden a disminuir los desfases de tiempo y presupuesto en este tipo de proyectos, sin basarse en la disminución de la calidad de vida de los profesionales involucrados, contribuyendo a que las empresas nacionales sean mucho más productivas y cuenten con mayor capacidad de competir internacionalmente.

Objetivos de la Investigación

General

- Documentar y difundir las buenas prácticas de gerenciamiento de proyectos de software aplicadas en el Ecuador, que estén orientadas a mejorar las decisiones gerenciales que ocasionan los desfases en tiempo y presupuesto para este tipo de proyectos.

Específicos

- Favorecer el crecimiento de la industria de software en el Ecuador, a través de las mejoras de productividad y competitividad de las empresas.
- Disminuir los niveles de tensión soportados por los equipos de desarrollo de software, ocasionados principalmente por las limitaciones de tiempo.
- Establecer una línea de base para siguientes estudios sobre el tema.

Hipótesis

- La mayoría de los desfases causados en tiempo y presupuesto en proyectos de desarrollo de software son causados por el uso de malas prácticas de planificación y manejo interno de los equipos de desarrollo.
- Los atrasos en proyectos de software pueden reducirse con la aplicación de ‘recetas prácticas’ de gerencia, utilizadas en proyectos llevados a cabo por empresas productoras de software ecuatorianas.

Metodología

Para el desarrollo del proyecto, se ha considerado seguir con los siguientes pasos de ejecución:

- Implementación de un sitio web en línea que permita realizar una encuesta electrónica, de tal manera que sea posible su difusión por medio de correo electrónico y, así llegar a la mayor cantidad de personas posibles.
- Luego de determinar el tamaño de la muestra estadística, se aplicará la encuesta a los empleados de empresas desarrolladoras de software en diferentes ciudades del país, tanto a niveles gerenciales como a niveles operativos, para determinar sus actuales prácticas en la planificación y distribución de tiempos, así como la estructura de costos en sus proyectos de desarrollo.
- Consolidación de la información y análisis estadístico.
- Entrevistas a profundidad con gerentes allegados, de ser posible llegar a entrevistas con los líderes de la AESOFT (Asociación Ecuatoriana del Software), MachángaraSoft (parque tecnológico), Colegios Profesionales y otros, para validar la información y la discusión de posibles soluciones.
- Discriminar las mejores prácticas utilizadas en proyectos exitosos y los puntos críticos de mejora, y proponer y contrastar con los posibles escenarios de aplicación.

Capitulo 2. Fundamentación teórica

En las últimas dos décadas, el software ecuatoriano logró un alto reconocimiento mundial y una gran clientela en el mercado local e internacional, por su alta calidad y adaptabilidad a diversos entornos. Muchas empresas multinacionales establecieron contactos con empresas de desarrollo de software ecuatorianas, resultando por ejemplo proyectos como la implementación de componentes dentro de las herramientas Office de Microsoft, en la Armada de los Estados Unidos y en muchos bancos en toda América. (Carmel, 2000)

En la actualidad, este mercado está en pleno crecimiento, inclusive luego de que el "momento de fama internacional" que tuvo el Ecuador en esta rama se dio aproximadamente entre los años 1995 y 2003. Los empresarios ahora están concentrados en consolidar esfuerzos para no perder el terreno ganado, y crear productos que puedan ser comercializados en mercados internacionales. (Martinez, 2009)

En base a este escenario se tomarán como marco teórico del presente estudio a las siguientes tendencias teóricas:

Software Propietario versus Software Libre

En la industria de software existen dos líneas de pensamiento que orientan la filosofía de desarrollo de productos, sus métodos de comercialización y sus mecanismos de distribución: Software Propietario y Software Libre. Los

modelos de negocio varían dependiendo de la tendencia que elija la empresa de software, y proporcionando diferentes retos de gerenciamento de acuerdo la filosofía seleccionada. En adelante se explica detalladamente cada una de ellas:

Software Libre**:**

El software libre es una filosofía reciente en el mercado mundial de software, y sus creadores conceptuales[10] lo relacionan directamente con el ejercicio de 'libertades' al respecto del acceso a la tecnología. Su objetivo principal es el de promover que el software sea publicado con su código fuente y puesto al alcance del público en sitios de Internet especializados, de tal manera que la comunidad mundial en general pueda beneficiarse de él (usarlo, estudiarlo, cambiarlo, publicarlo e inclusive comercializarlo).

Dentro de ésta línea de pensamiento existen las siguientes sub-especializaciones:

- *Código Abierto (Open Source)*[11]: Sus promotores se refieren a ésta filosofía no solamente como la posibilidad de acceso al código fuente de una aplicación, sino el del cumplimiento de diversos criterios que dan forma a esta filosofía: Libre distribución, acceso al código fuente, distribución de trabajos derivados, Integridad sobre el trabajo original del autor, evitar discriminación, libertad de uso en cualquier entorno, licencia

[10] Aunque hay diversos actores en esta línea de pensamiento, generalmente se refiere como iniciador de este movimiento a Richard Stallman, creador del concepto de software libre y de la fundación 'Free Software Foundation'. (Wikipedia, Software Libre, 2011)
[11] Se atribuye el acuñamiento del término y la filosofía a Eric Raymond y Bruce Perens, que formaron la Open Source Initiative (OSI).

de distribución, tecnología neutral y otras (Open Source Initiative, 2011). Las herramientas de código abierto no necesariamente implican que están libres de costo.

- *Software Libre (Free Software)*: La traducción del término “Free” en el idioma inglés parecería presentarse como software gratuito, pero en realidad a lo que se refiere es a las libertades y no a la gratuidad. Su concepto se fundamenta en el cumplimiento de libertades y fundamentos morales y éticos al respecto de los productos de software. Las principales libertades corresponden a la capacidad de ejecutarlo, de verificar internamente su funcionamiento, copiarlo, estudiarlo, modificarlo y distribuirlo, lo que implica en el contar con el código fuente. A pesar que sus conceptos son muy parecidos con respecto al Código Abierto (Open Source), su diferencia radica en que ésta última promueve la importancia práctica de compartir el código fuente, mientras que el Software Libre promueve los principios de libertad y ética al respecto del software.

  De acuerdo con tal definición, un software es "libre" cuando garantiza las siguientes libertades: (Wikipedia, Software Libre, 2011)

Libertad	Descripción
0	la libertad de usar el programa, con cualquier propósito.
1	la libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.

2	la libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
3	la libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.
Las libertades 1 y 3 requieren acceso al código fuente porque estudiar y modificar software sin su código fuente es muy poco viable.	

- *Software Gratuito (Freeware)*: Se refiere a software que normalmente se encuentra disponible de forma 'gratuita', aunque en muchos casos con ciertas restricciones de uso o de acceso al código fuente. (Wikipedia, Freeware, 2011)
- *Software de dominio público*: Es un software que no requiere licencia, pues sus derechos de explotación son para toda la humanidad, cualquier puede hacer uso de él con fines legales y consignando su autoría original. Este software correspondería al que es donado a la humanidad. (Wikipedia, Software Libre, 2011)

Una vez que un producto de software libre ha empezado a circular, rápidamente está disponible a un costo muy bajo. Al mismo tiempo, su utilidad no decrece. El software, en general, podría ser considerado un bien de uso inagotable, tomando en cuenta que su costo marginal es pequeñísimo y que no es un bien sujeto a rivalidad (la posesión del bien por un agente económico no impide que otro lo posea).

Puesto que el software libre permite el libre uso, modificación y redistribución, a menudo encuentra un hogar entre usuarios para los cuales el coste del software no libre es a veces prohibitivo, o como alternativa a la piratería. También es sencillo modificarlo localmente, lo que permite que sean posibles los esfuerzos de traducción a idiomas que no son necesariamente rentables comercialmente.

La mayoría del software libre se produce por equipos internacionales que cooperan a través de la libre asociación. Los equipos están típicamente compuestos por individuos con una amplia variedad de motivaciones, y pueden provenir tanto del sector privado, del sector voluntario o del sector público. Existen muchos criterios acerca de la relación entre el software libre y el actual sistema político-económico:

- Algunos consideran el software libre como un competidor contra el centralismo[12] en empresas y gobiernos, una forma de orden espontáneo o de anarquismo práctico.
- Algunos consideran el software libre como una forma de trabajo colaborativo en un modelo de mercado, tal como se había planteado el cooperativismo[13].

[12] El centralismo es el sistema de organización estatal cuyas decisiones de gobierno son únicas y emanan de un mismo centro, sin tener en cuenta las diferentes culturas o pueblos a quienes afecta. (Wikipedia, Centralismo, 2011)
[13] El cooperativismo es una doctrina socio-económica que promueve la organización de personas para sarisfacer, de manera conjunta sus necesidades. (Liga Coop)

- Algunos comparan el software libre a una economía del regalo[14], donde el valor de una persona está basado en lo que ésta da a los demás, sin que incurra valor monetario formal de por medio.

Las implicaciones políticas y económicas del software libre y su afinidad con el anti-autoritarismo son discutibles. Mientras para unos estas implicaciones son notorias y representan un factor importante a tomarse en cuenta, para otros si bien podría existir una leve relación, no tiene suficiente relevancia.

El negocio detrás del software libre se caracteriza por la oferta de servicios adicionales al software como: la personalización y/o instalación del mismo, soporte técnico, donaciones, patrocinios; en contraposición al modelo de negocio basado en licencias predominante en el software de código cerrado.

El mayor representante de ésta filosofía es el sistema operativo llamado Linux, creado por Linus Torvalds, el cuál es distribuido abiertamente con su código fuente en múltiples versiones mantenidas por sus usuarios. Existe un catálogo de software libre en diversas páginas como la Wikipedia (http://en.wikipedia.org/wiki/Free_Software_Portal) o Source Forge (http://sourceforge.net/). Es posible desarrollar software libre también con herramientas propietarias como es el caso de CodePlex (http://www.codeplex.com), que promueve el desarrollo de herramientas de código abierto bajo tecnología Microsoft.

En el Ecuador existe la Asociación Ecuatoriana de Software Libre (http://www.asle.ec/), que son los encargados de promover esta filosofía en el

[14] La economía del regalo, o también llamada la economía del don, se basa en el principio de vivir bajo la premisa de que "a mi vecino no le falte nada". (Wikipedia, Economía del don, 2011)

país, desarrollando múltiples eventos y encuentros, alineados con las tendencias dadas a nivel mundial.

El gobierno ecuatoriano, a través del decreto ejecutivo Nro. 1014, determinó la importancia del uso de software libre como política de estado, de tal manera que se oficialice su uso en las instituciones públicas como un estándar del poder ejecutivo (Decretos Presidenciales, 2008).

Software Propietario**:**

Son herramientas que se reservan la publicación del código fuente, y su negocio consiste en la provisión de las soluciones a los clientes en base a un pago por licencia de uso.

En este caso el negocio del software se concentra específicamente en la venta de licencias, que comúnmente implica la provisión de servicios, desarrollo de nuevas versiones y soporte constante.

Los clientes tienen mayor certeza al respecto de los costos de implementación, ya que se miden únicamente en función de los valores de las licencias, ya que en el caso del Software Libre existen múltiples costos ocultos durante los procesos de implementación con los usuarios finales[15].

El mayor representante de esta línea de pensamiento es el sistema operativo Windows de la empresa Microsoft, que se ha convertido en el software más popular en el mercado, pero a su vez es uno de los programas que ha sufrido

[15] Los costos ocultos existen por el hecho de que la mayoría del software libre, no equivale a software gratuito como se pensaría en base a su término. Este tipo de software tiene costo por las licencias corporativas (las licencias públicas básicas suelen ser gratuitas), costo por el soporte, costo por la personalización, costo por la implementación, etc.

en mayor intensidad el efecto de la piratería[16]. En el mercado existen grandes empresas que trabajan con software propietario, como es el caso de Microsoft, IBM, Oracle, Adobe, Sybase, etc.

En el mercado existen miles de empresas que trabajan con esta filosofía de desarrollo de proyectos de software, presentando soluciones de toda índole basándose en el ejercicio completo de los derechos de autor, reserva de su código fuente como secreto empresarial, cobro por licencias de uso y limitaciones en diversos ámbitos al respecto de su distribución. Con esta filosofía nació el negocio del software y todas sus variaciones actuales.

La presente investigación tratará de diferenciar los comportamientos de planificación para ambas tendencias expuestas, buscando determinar si la elección tomada por las empresas al respecto de estas orientaciones, constituye una variable determinante para que existan desfases y atrasos en el desarrollo del software.

Predictibilidad

Existen dos líneas de pensamiento al respecto de los métodos de planificación a construir un software, estos son los siguientes:

[16] Es habitual el uso del término piratería (Infracción de derechos de autor), para referirse a las copias de obras sin el consentimiento del titular de los derechos de autor y sin pago de licencias de algún tipo. El físico Richard Stallman y el experto en propiedad intelectual, Eduardo Samán, entre otros, argumentan que el uso de la expresión piratería para referir a las copias no autorizadas es una exageración que pretende equiparar el acto de 'compartir' con la violencia de los piratas de barcos, criminalizando a los usuarios. (Wikipedia, Infracción de derechos de autor, 2011)

Software no predecible:

Esta línea de pensamiento toma como premisa que el software no es un producto fácilmente delimitable, sino más bien un proceso en constante evolución, por lo que afirma que la cantidad de variaciones que pueden presentarse sobre el diseño del software no son predecibles, por ello la planificación se basará principalmente en la experiencia del planificador y en base a eso calcular márgenes de riesgo al respecto del tiempo y el presupuesto, para soportar los porcentajes de variación acostumbrados en el equipo de trabajo.

Software predecible:

Esta línea de pensamiento afirma que las variaciones sobre el diseño se pueden medir y cuantificar, de tal manera que sea posible establecer un algoritmo o fórmula matemática que permita aproximarse con suficiente exactitud a la cuantificación del tiempo y presupuesto del proyecto.

Los modelos internacionales como el modelo COCOMO, Estimación Paramétrica, Modelo Putman, SEEM SER, Análisis de Puntos funcionales, etc., generalmente trabajan sobre esta línea del pensamiento, a través de un conjunto de fórmulas matemáticas que ofrecen entregar una aproximación que puede utilizarse de forma confiable durante la planificación.

La presente investigación se orientará al lineamiento de que el software no es predecible, en vista de que esa orientación parece ser la más cercana a la realidad del desarrollo de software en el Ecuador. Esta afirmación será

corroborada por medio de las encuestas y entrevistas a profundidad previstas en el presente estudio.

Arquitectura del Software

La arquitectura de software es una tarea técnica realizada por un especialista que generalmente cuenta mucha experiencia en desarrollo de software. Su tarea es entender las funcionalidades que deberá implementar el sistema, diseñar un escenario especializado, y plantear una estructura tecnológica que facilite la etapa de desarrollo del software. Existen dos líneas de pensamiento al respecto de este tema:

La arquitectura de software es trascendental**,** En esta línea de pensamiento se da mucha importancia a las formas en las cuales el arquitecto utiliza la tecnología existente para orientar el inicio de la solución de software y asegurar su éxito tecnológico y operacional.

La arquitectura de software no es lo importante, lo importante es la lógica de negocio**.** En esta línea de pensamiento no se da mucha importancia a las tareas de arquitectura, sino que se prefiere dedicar mucho mayor tiempo a las tareas relacionadas con la programación del software. Para solventar los problemas tecnológicos, se busca en Internet soluciones previamente hechas o se encarga a uno de los desarrolladores del equipo la tarea. Normalmente en este tipo de equipos de trabajo se busca que el especialista sea la persona que conoce el giro del negocio que el software deberá implementar.

Lastimosamente en el país el título de “arquitecto de software” no es otorgado por ninguna institución que valide los conocimientos o experiencia, por lo que

mucha gente se auto-titula de esa manera, con el fin de lograr mejores condiciones laborales.

El presente estudio se orientará sobre la línea de pensamiento de que la arquitectura de software es trascendental para el éxito del proyecto, en vista de que con una buena estructuración tecnológica al inicio, acompañada de una definición clara al respecto de los componentes que intervendrán en el desarrollo, es posible disminuir los desfases de tiempo y presupuesto. Se tratará de corroborar estos supuestos por medio de las encuestas y análisis posterior.

Capitulo 3. Estado del Arte

Varias de las grandes empresas de tecnología, universidades y organizaciones de establecimiento de estándares, tales como Microsoft, IBM, Oracle, MIT, ISO y otros, han propuesto un sinnúmero de metodologías que buscan establecer un estándar común para lograr el éxito de los proyectos de desarrollo del software y para asegurar su cumplimiento "a tiempo".

En adelante se realizará un abordaje teórico sobre las metodologías más importantes que han sido utilizadas en diversos países y bajo diferentes entornos, de tal manera que sirvan de línea de base para una comparación posterior al respecto de lo utilizado en el ámbito ecuatoriano:

Microsoft ® Solutions Framework (MSF)[17]

Es un enfoque disciplinado para proyectos de tecnología, basada en un conjunto definido de principios, modelos, disciplinas, conceptos, orientaciones y prácticas, que son aprovechados durante la ejecución de proyectos de desarrollo de software. MSF proporciona herramientas para lograr una visión general

Figura 1 Ciclo conceptual para desarrollo de software propuesto por MSF

[17] En la empresa en donde se ha tenido la oportunidad de generar diversas experiencias en el ámbito del desarrollo de software, se ha utilizado principalmente la metodología MSF, ya que es la que se ha conocido con profundidad y se ha encontrado mejor adaptabilidad en proyectos ecuatorianos. Es común el uso de esta tecnología en empresas que utilizan la tecnología propuesta por la empresa Microsoft.

del proyecto, los modelos básicos y las disciplinas esenciales, centrándose en cómo su uso contribuye al éxito de los proyectos de tecnología. MSF contrasta con las metodologías y estándares de la industria y describe cómo puede ser utilizado en combinación con ellos.

El ciclo de desarrollo de software según esta metodología cumple 5 etapas claramente definidas: el visionamiento, la planificación, el desarrollo, la estabilización y la instalación, como se muestra en el siguiente gráfico:

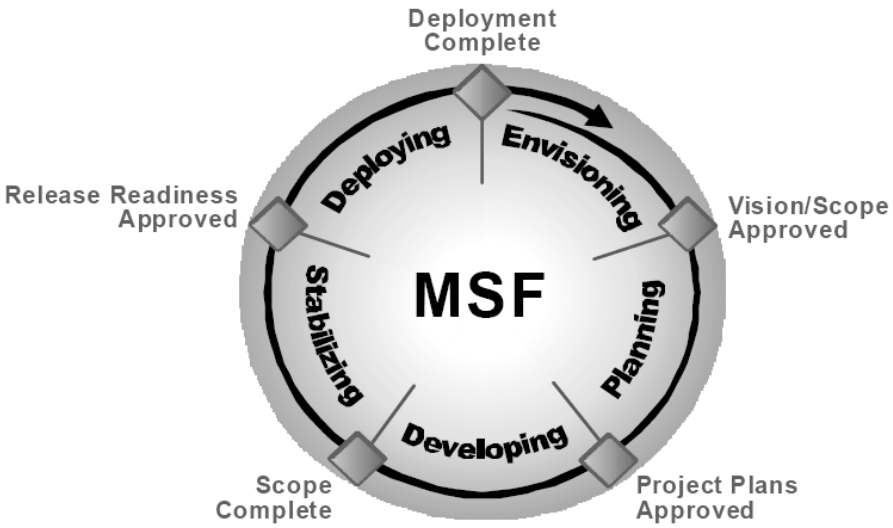


Figura 2 Pasos para el desarrollo de software propuesto por MSF

MSF provee para cada una de las etapas un conjunto de plantillas de análisis y formatos estándar de documentación, que permiten organizar la información resultante y con ella generar documentación técnica fundamental para los equipos de desarrollo. (Microsoft, 2003)

El visionamiento consiste en la definición de los límites globales del proyecto, facilitando la detección de los objetivos prioritarios, los requerimientos fundamentales, los riesgos inherentes y los actores que participarán en el desarrollo.

La planificación o también llamada etapa de análisis consiste en estudiar uno por uno los procesos de forma detallada basándose en entrevistas con los actores, de tal forma que se puedan crear los modelos y diagramas del flujo de la información, y llegar a la especificación final del sistema.

El desarrollo consiste en la implementación de los estudios realizados en el visionamiento y la planificación, a través de una herramienta de programación.

La estabilización consiste en la verificación de que los objetivos planteados se hayan cumplido en el desarrollo de la aplicación asegurando la calidad del mismo.

La instalación o puesta a producción consiste en colocar a disposición de los usuarios finales los servicios y las aplicaciones desarrolladas.

Este ciclo es repetido las veces que sean necesarios para cumplir con todas las etapas y versiones del sistema.

CMMI (Capacity Maturity Model Integration)

Capability Maturity Model Integration (CMMI) es un enfoque de mejora de procesos que ayuda a las organizaciones a mejorar su desempeño. CMMI

puede ser usado para guiar la mejora de procesos a través de un proyecto, una división o una organización entera.

CMMI en ingeniería de software y el desarrollo organizacional es un enfoque de mejora de procesos que proporciona a las organizaciones los elementos esenciales para la mejora de procesos eficaces. CMMI está registrado en la Oficina de Patentes y Marcas de EE.UU. por la Carnegie Mellon University.

De acuerdo con el Software Engineering Institute (SEI, 2008), CMMI ayuda a "integrar las funciones de organización tradicionalmente separadas, establecer objetivos de mejora de procesos y prioridades, proporcionar una guía para los procesos de calidad, y proporcionar un punto de referencia para evaluar los procesos actuales." (Wikipedia, Capability Maturity Model Integration, 2011)

CMMI define las prácticas que las empresas han implementado en su camino hacia el éxito. Estas prácticas abarcan temas que incluyen obtener y gestionar los requisitos, toma de decisiones, la medición del rendimiento, planificación del trabajo, manejo de riesgos, y mucho más.

El uso de estas prácticas busca mejorar las probabilidades de éxito en los negocios. Las prácticas del CMMI se pueden utilizar en un equipo, un grupo de trabajo, un proyecto, una división o una organización entera. (Carnegie Mellon University, 2011)

El proceso de implementación de CMMI implica el cubrir los siguientes pasos (Carnegie Mellon University, 2011):

- Aseguramiento del Patrocinio y Financiación: Antes de comenzar el esfuerzo de mejora de procesos, es importante asegurarse que su programa de mejora de procesos tiene un patrocinador que forma parte de la alta dirección de la empresa y proporcionará el financiamiento. Este patrocinio y financiación es fundamental para garantizar el éxito del programa.
- Tomar información base: Es importante comprender los conceptos básicos de CMMI. Para este efecto existen en Internet varios programas de introducción y cursos de introducción a la metodología de implementación.
- Preparar la organización para el cambio: Es necesario tratar la mejora de procesos como un proyecto, establecer las razones de negocio y los objetivos del negocio para justificar el esfuerzo. Crear un argumento convincente para el cambio, incluyendo la justificación de la empresa y los beneficios y costos esperados para los involucrados. Desarrollar una presentación persuasiva de los problemas y oportunidades. Las personas clave involucradas también deben tener el entrenamiento básico.
- Formar un Grupo de Orientación del Proceso: Este grupo coordina las actividades de mejora de procesos en la empresa. Los miembros del grupo puede servir como mentores de mejora de procesos. Si el grupo de procesos es nuevo para la mejora de procesos, los miembros deberían considerar la posibilidad de tomar los cursos avanzados sobre la metodología.

- Saber dónde estamos: utilizar las mejores prácticas CMMI para realizar un mapa de los procesos de la organización y hacer un análisis de las deficiencias informales (por ejemplo, SCAMPI C) para determinar en qué nivel se encuentra la empresa con las prácticas del modelo CMMI, y verificar el cumplimiento de la clase C del método de evaluación[18]. Hacer una encuesta para recopilar datos de los administradores, líderes de proyecto, y los trabajadores para evaluar las oportunidades y las barreras culturales al cambio. Construir una imagen detallada de la actualidad.
- Saber hacia dónde dirigirnos: Forjando en el imaginario una imagen de un lugar, es necesario crear una imagen del lugar donde se desea que la empresa esté. Caracterizar el éxito que se desea antes de empezar. Obtener una visión equilibrada de la administración, jefes de proyecto, y el personal acerca de lo que ellos piensan que es más importante. Cada uno tendrá diferentes objetivos que quieren alcanzar. Dar prioridad a las áreas de proceso para abordar y construir su plan de mejora. Seguimiento de su progreso contra el plan.
- Comunicar y coordinar: Usted debe tener una comunicación abierta y honesta. Comparta el plan con todos los que se verán involucrados y escuchar sus comentarios.
- Seguimiento del progreso: Comparar la imagen de dónde se encontraba la empresa al inicio, con la que se desea ser. La diferencia entre ambas

[18] SCAMPI es un método de evaluación de la implementación de CMMI y tiene diferentes clases de evaluación A, B y C. Estos procesos son llevados a cabo por consultores que determinan el cumplimiento de los estándares y presentan un informe. La clase C al ser la más rápida de realizar, proporciona un buen punto de arranque para el establecimiento de tareas del CMMI antes de arrancar con el proyecto. (Process Strategies Inc., 2010)

imágenes es el enfoque de su programa de mejora de procesos. Crear un informe periódico (por ejemplo, mensual, semanal) que demuestre el progreso del programa para llegar a sus objetivos (y de la organización). Usted también puede contar con un evaluador autorizado que lleve a cabo una consultoría, que proporcionará una evaluación objetiva de su organización utilizando el método SCAMPI.

Microsoft realizó un convenio de uso de la metodología CMMI creada por la Universidad Carnegie Mellon para la integración con su metodología de desarrollo de software MSF, forjando una especialización llamada "MSF for CMMI Process Improvement". Adicionalmente desarrolló herramientas que faciliten a los equipos de desarrollo la aplicación de la metodología.[19]

Microsoft propone para su metodología el involucramiento de personas que cubren roles, los roles realizan actividades, las cuales están agrupadas en diversas líneas de trabajo, las actividades producen productos, los cuales tienen que pasar por ciertos procesos para ser elaborados. Un producto puede ser un documento, hojas de cálculo, planes de proyectos, código fuente y otros.

La definición del producto, el desarrollo y las pruebas del software se realizan utilizando múltiples iteraciones, dando lugar a la terminación gradual del proyecto.

[19] Microsoft tiene un producto llamado Microsoft Visual Studio Team System, que consiste en el uso de un servidor llamado Team Foundation Server, que permite que todos los programadores se conecten a él y realicen sus tareas de forma coordinada. Dentro de este producto es posible establecer y ejecutar la metodología de análisis y filosofía de trabajo elegida. Por ejemplo existen plantillas para CMMI, para Agile, SCRUM y otros.

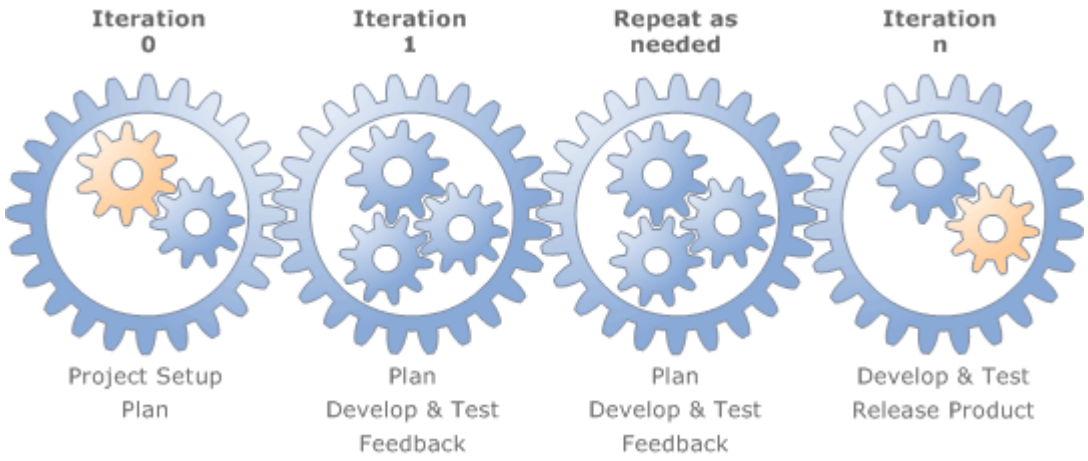


Figura 3: Modelo de Iteraciones de un proyecto de software

La ventaja de contar con Iteraciones pequeñas es que permiten reducir el margen de error en sus estimaciones y proporcionar información rápida acerca de la exactitud de los planes de su proyecto.

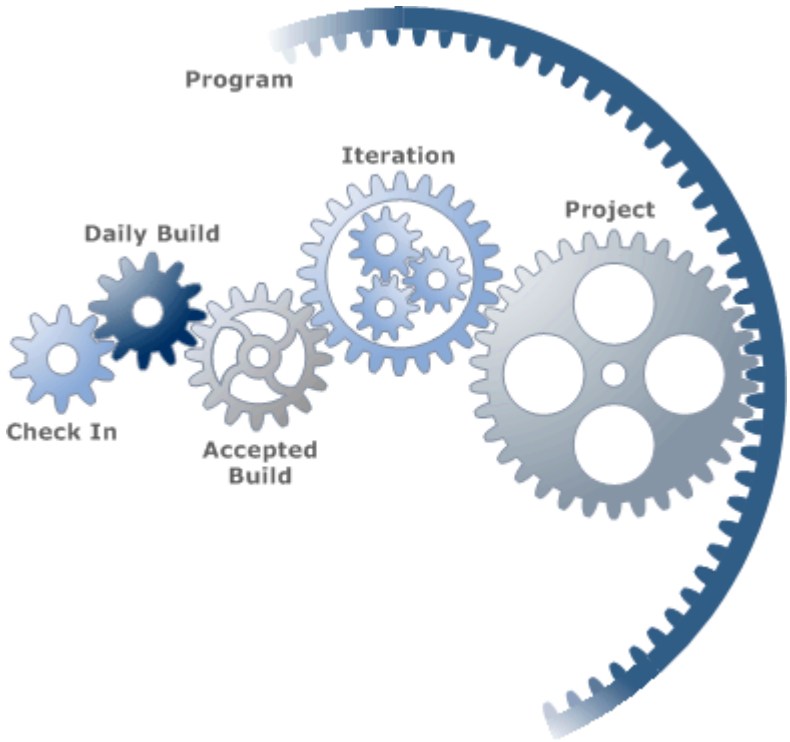


Figura 4: Modelo de Iteraciones de CMMI

Gobernabilidad: La gobernabilidad alude a la utilización óptima de los recursos a través del control de tiempo y dinero en relación con la cadena de valor.

"MSF for CMMI" define cinco líneas para el ciclo de vida del proyecto de software, alineado con la metodología MSF, que encapsulan flujos de trabajo y actividades. Cada punto de control ofrece la oportunidad de autorizar la continuidad del trabajo en el proyecto o suspenderlo en caso de detectar tempranamente anomalías o puntos de divergencia. Un punto de control en cada momento hace una pregunta diferente al respecto de la ejecución del proyecto, y el objetivo de la gobernabilidad es el contar con respuestas a estas preguntas por medio de la información obtenida de las operaciones del día a día de la ingeniería de software.

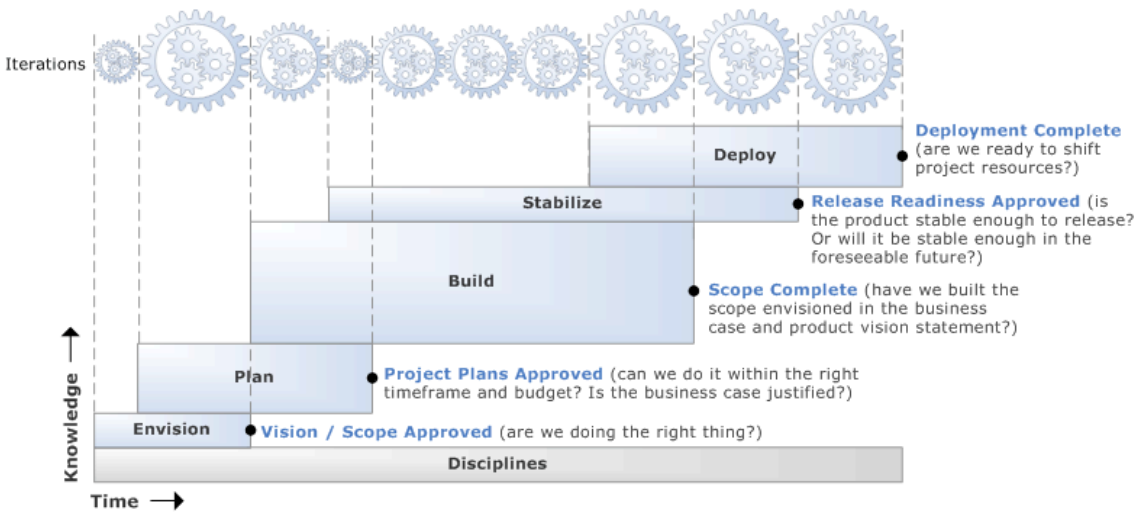


Figura 5: Explicación del modelo iterativo dado en la fusión de MSF y CMMI

El Modelo de Gobierno de MSF está diseñado para ser tolerantes a los riesgos y facilitar la circulación fluida de un proyecto. El riesgo residual se equilibra con el deseo de mantener el proyecto en movimiento y ser a la vez ágiles y productivos dentro de un marco de gobierno formal. La idea a la larga es separar la gestión operativa del proyecto de software de la organización del gobierno corporativo de la organización. Por eso durante el planteamiento de la metodología CMMI para el desarrollo de software, hay que responder si somos

buenos en ingeniería de software y si estamos haciendo el mejor uso de nuestros recursos de ingeniería de software.

Después de todo, la metodología busca que la gestión operativa se concentre en la capacidad, calidad y fiabilidad del software. La gobernabilidad es acerca de hacer el mejor uso de los fondos de los accionistas o de los contribuyentes a través de la utilización óptima de la capacidad disponible. (Microsoft, 2005)

Desarrollo de software Ágil

Es un método iterativo para determinar las necesidades de software y para la entrega de los proyectos de una manera muy flexible e interactiva. Se requiere el empoderamiento a los individuos de la empresa, con los proveedores y los requerimientos de los clientes. También hay enlaces metodológicos con las técnicas Lean y 6-Sigma. Las técnicas ágiles son más utilizadas sobre proyectos de pequeña escala o para componentes que forman un programa más amplio de trabajo.

Las técnicas ágiles también pueden ser llamadas "Administración de Proyectos Extrema"[20]. Es una variante del ciclo de vida iterativo[21], donde las prestaciones se presentan en etapas. Una diferencia entre ágil y desarrollo iterativo es que el tiempo de entrega de Agile es en semanas y no en meses. Desde la metodología de desarrollo de proyectos Ágil, se deriva del desarrollo ágil de

[20] Para búsquedas en Internet, se lo hace referencia en Inglés como Extreme Project Management

[21] Este método del ciclo de vida iterativo tradicional ha sido usado durante décadas y ha demostrado su efectividad de aplicación. De hecho, el Departamento de Defensa de EE.UU. está promoviendo activamente el uso de este método en todos sus proyectos cuando publicó el estándar 2167A en 1998. Se define como un modelo de desarrollo secuencial con resultados claramente definidos para cada fase. Muchos profesionales de la industria son estrictos en la realización de exámenes de auditoría para asegurar que el proyecto ha cumplido con los criterios de entrada antes de continuar a la siguiente fase. (Executive Brief, 2008) En inglés se lo conoce comúnmente como "Waterfall" o metodología de cascada y es la base conceptual para todas las metodologías de desarrollo de software.

software, sigue los mismos estándares definidos en el "Agile Manifiesto"[22] cuando se trata de la colaboración y la documentación. Existen varios métodos de software se derivan de Agile, que dentro de los más utilizados se encuentran SCRUM y Extreme Programming.

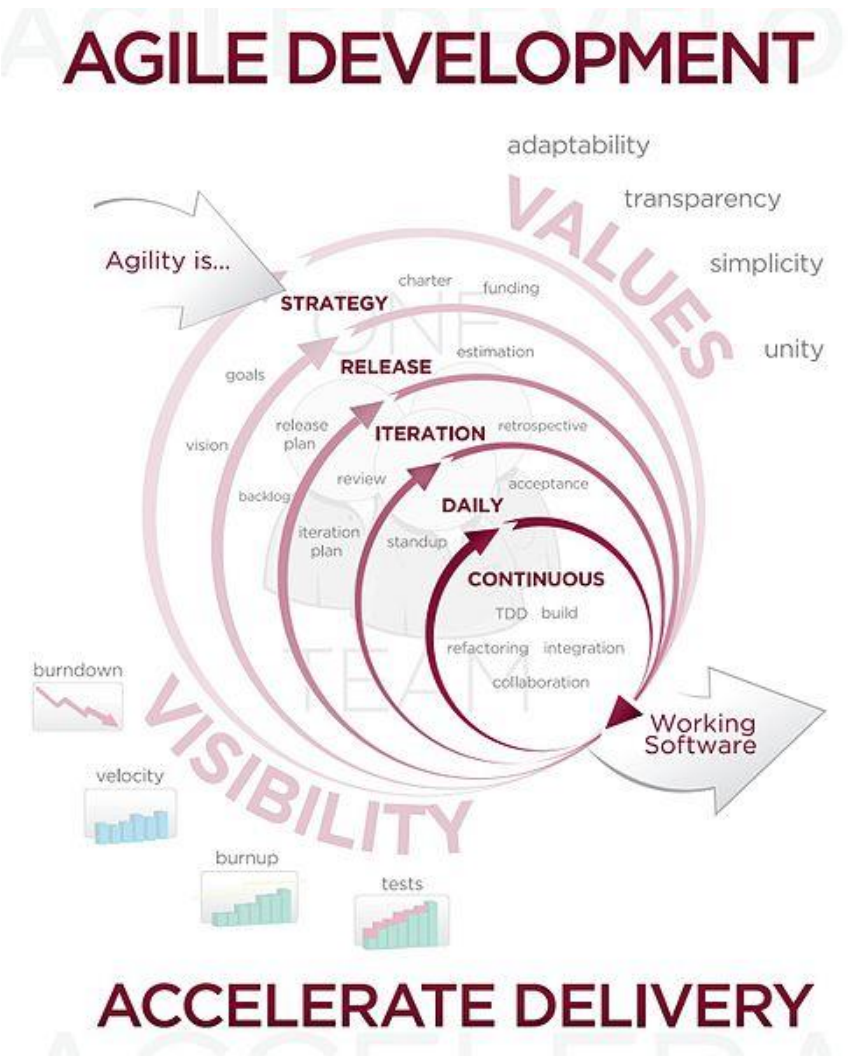


Figura 6: Diagrama de iteración de la Metodología Ágil

[22] Las definiciones conceptuales del "Agile Manifiesto" pueden encontrarse en la siguiente dirección web: http://en.wikipedia.org/wiki/Agile_Manifesto

La metodología ágil tiene sus desventajas: Muchos creen que no funciona bien en proyectos de grande escala, pues los proyectos de software de gran tamaño siguen utilizando la metodología en cascada. Además, la ventaja y utilidad de la metodología ágil radica en su aplicación sobre proyectos que tienen cambios frecuentes. El desarrollo ágil no ofrece ninguna ventaja sobre la metodología en cascada cuando se trata de proyectos clásicos donde los requisitos son casi siempre constantes y las variaciones son poco comunes, tales como proyectos de construcción. (Wikipedia, Agile Software Development, 2011)

SCRUM

SCRUM es un marco de trabajo iterativo e incremental para la administración de proyectos de software, descendiente del desarrollo de software ágil.

A pesar que la metodología SCRUM fue originalmente sugerida para el manejo de proyectos de desarrollo de productos, se uso se ha enfocado en el desarrollo de proyectos de software, y puede ser usado inclusive para administrar equipos de mantenimiento de software, o en general para la administración de proyectos.

SCRUM es un proceso “esqueleto” que contiene un conjunto de prácticas y roles predefinidos. Los roles más importantes son:

- ScrumMaster: quien administra los procesos (el que corresponde a Administrador del proyecto)
- Product Owner: Quien representa a los accionistas y al negocio

- Team: Un grupo multifuncional conformado por 7 personas, que realizan el proceso de análisis, diseño, implementación y pruebas del software.

El proceso consiste en que cada “sprint” o “iteración”, típicamente que toma un tiempo de dos a cuatro semanas, el equipo realiza la entrega de un producto final. Las características a ser implementadas en cada iteración son tomadas de la base de requerimientos realizada durante la etapa de planificación, priorizando en primer lugar a los requerimientos más importantes.

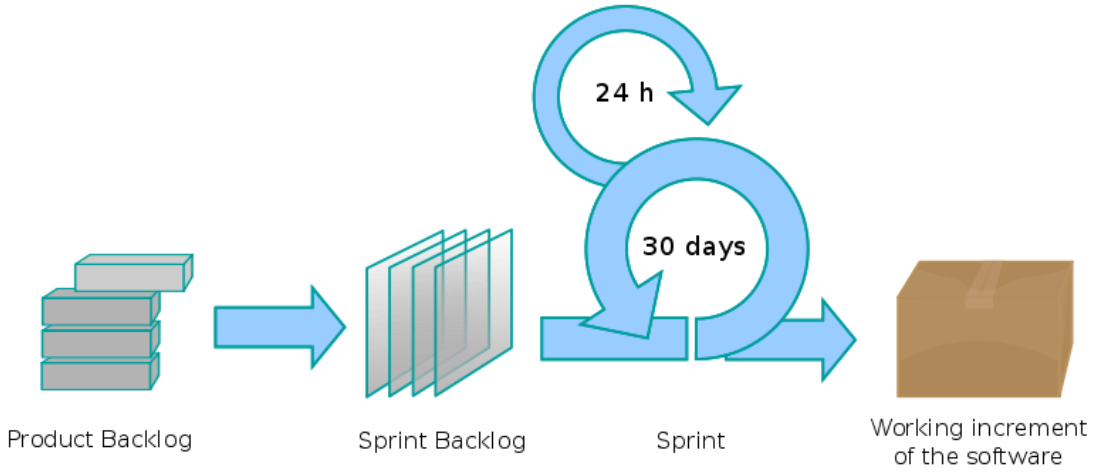


Figura 7: Modelo de Iteraciones de SCRUM

Para el inicio de cada iteración, se realiza una reunión con el Product Owner, quien valida y acuerda con el equipo las características a ser implementadas en ese producto entregable. No existen modificaciones a las características incluidas para ese entregable hasta que se termine la iteración.

La programación tiene una limitación de tiempo fija[23] para cada iteración y debe terminar obligatoriamente en ese tiempo. Si un requerimiento no pudo ser entregado por alguna razón, se lo dejará de lado y regresará al listado de requerimientos. Para la presentación de un entregable, es necesario que el equipo demuestre cómo usar el software.

La clave principal de SCRUM es que reconoce que durante el desarrollo del proyecto, los clientes / usuarios del software cambian su manera de pensar acerca de lo que quieren y necesitan. Esos cambios son normalmente imprevisibles conforme a la planificación predictiva tradicional. Scrum en este sentido adopta un enfoque empírico, aceptando que un software no puede ser completamente entendido o definido, concentrando los esfuerzos en maximizar la habilidad del equipo sobre la presentación ágil de entregables y ajustarse a los requerimientos cambiantes.

Scrum puede ser implementado por medio del uso de un amplio espectro de herramientas. La mayoría de empresas usan herramientas universales, como por ejemplo hojas de cálculo para llevar el listado de requerimientos y la información sobre las iteraciones. Existen varias herramientas de código libre, que permiten llevar la metodología Scrum para la construcción de productos. (Wikipedia, Scrum (development), 2011)

Extreme Programming[24]

Es una metodología de desarrollo de software que tiene el enfoque de mejorar la calidad de software y la agilidad de respuesta ante los requerimientos

[23] En la literatura se refiere a este tipo de limitaciones de tiempo como “TimeBoxing”
[24] Esta metodología en la literatura se lo abrevia como “XP”.

cambiantes de los clientes. En vista que es una subclase de la metodología de desarrollo ágil, encamina los esfuerzos en la obtención de entregas frecuentes en cortos tiempos, lo cual implica la mejora en productividad e introduce puntos de control en donde el cliente puede incorporar cambios.

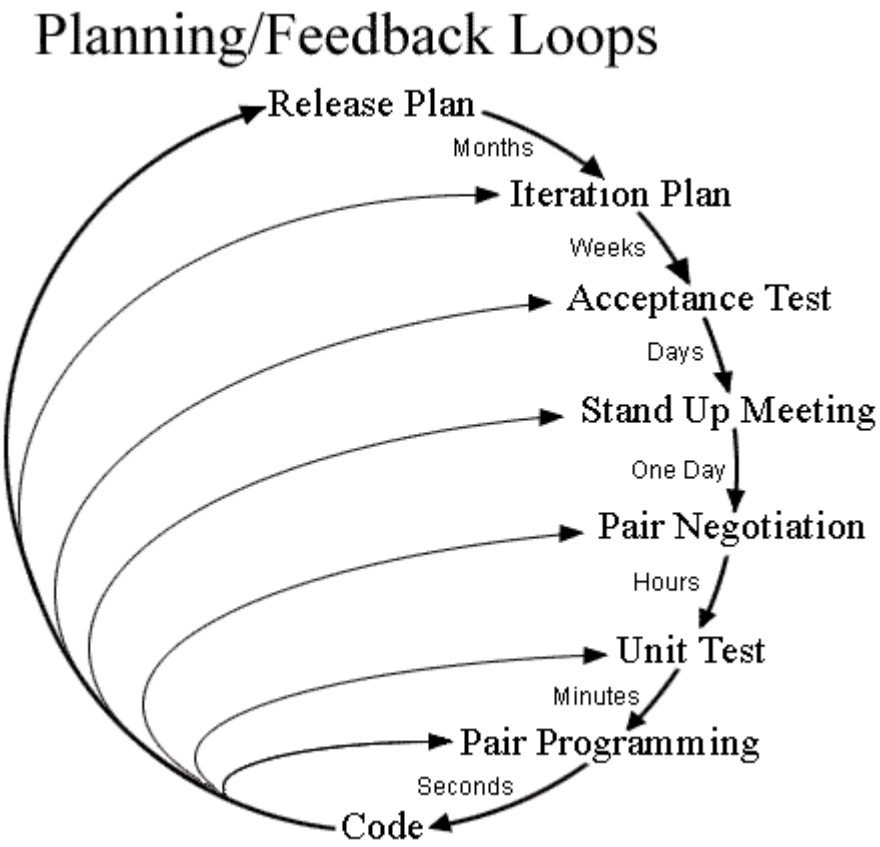


Figura 8 Ciclo de Iteración de la Metodología Extreme Programming

Extreme Programming es la disciplina que organiza a las personas para obtener un software más productivo, bajo la premisa que se reducirán los costos de los cambios, al hacer iteraciones de corta duración. Presenta como doctrina que los cambios en requerimientos son naturales, inevitables y es un aspecto indiscutible de los proyectos de desarrollo de software, considerando que dichos cambios deben ser planificados, en lugar que tratar de hacer una lista de requerimientos estable.

Extreme Programming define cuatro actividades básicas que deben ser llevadas a cabo como parte de un proceso de desarrollo de software: Codificando, Probando, Escuchando, Diseñando:

Codificando

Los defensores de Extreme Programming señalan que el producto realmente importante durante un proyecto de desarrollo de software es el código fuente. Sin código fuente, no hay producto.

El código sirve para expresar los pensamientos y problemas que enfrenta el programador, inclusive para demostrar la complejidad que debe ser implementada y que es difícil de explicar a terceros. Una vez que se cuenta con el código fuente, servirá para proponer una posición específica de resolución del problema, es claro y conciso y puede ser interpretado solamente de una forma. Otros programadores pueden leer el código fuente y proponer mejoras.

Probando

El enfoque de Extreme Programming señala que si se realizan pocas pruebas, se eliminarán ciertos defectos en el software. Si se realizan muchas pruebas, es posible eliminar muchos defectos.

Para este efecto se utilizan las pruebas de unidad, que son pruebas automatizadas realizadas por el mismo programador. Si al ejecutar la prueba, el resultado es satisfactorio, entonces puede decirse que el código se encuentra

completo[25]. También se utilizan las pruebas de aceptación, para verificar que los requerimientos fueron entendidos por los programadores para satisfacer los requerimientos actuales de los clientes.

Escuchando

Los programadores deben escuchar lo que los clientes necesitan realmente que el software haga, qué lógica de negocio es requerida. Deben entender estos requerimientos tan bien como para que sea posible que propongan retroalimentación acerca de los aspectos técnicos de cómo el problema puede ser resuelto o no resuelto. La comunicación entre el cliente y el programador se lo realiza por medio de una metodología llamada "Planning Game"[26].

Diseñando

Desde el punto de vista de la simplicidad, parecería ser que con el hecho de codificar, probar y escuchar, ya se lograría obtener software funcionando. En la práctica no funciona así. No es posible mirar al largo plazo sin una etapa de diseño en algún punto específico. El sistema se vuelve complejo y las dependencias internas ya no son claras. Uno evita estos problemas bajo el establecimiento de una estructura que organiza la lógica del sistema. Un buen diseño evitará las interdependencias, significando que si se realiza un cambio

[25] Para cumplir esta etapa, se organiza internamente un "Testathlon" que es un evento en donde todos los programadores realizan la escritura de código de pruebas de forma colaborativa, como una especie de lluvia de ideas al respecto de la forma como automatizar las pruebas.

[26] Esta metodología se realiza una vez en cada iteración o bajo necesidad, y consiste en realizar dos etapas: la primera implica una reunión con el cliente para determinar cuáles requerimientos deben ser incorporados inmediatamente y cuáles a continuación, y la segunda etapa consiste en realizar una reunión interna en el equipo de programadores para planificar las tareas que se requieren hacer para satisfacer los requerimientos. El propósito de esta metodología no es el establecimiento de fechas fijas, sino el proporcionar una orientación hacia el avance y entrega de productos.

en una parte del sistema, otras partes no serán afectadas. (Wikipedia, Extreme Programming, 2011)

Una de las prácticas más novedosas al respecto de la metodología Extreme Programming y en general de las técnicas ágiles es la llamada “Pair Programming” o programación en pares. Consiste en que en una misma computadora se encuentran dos programadores a la vez. El primero es llamado “driver” quien es el que realiza la programación del sistema. El otro es llamado “observer”, quien va revisando cada línea de código escrita por su compañero, proporcionando retroalimentación inmediata, proporcionando una alta reducción de errores a partir de la fuente.

Figura 9: Representación de la programación en pares

Estudios han revelado que esta técnica ha llegado a reducir entre el 15% y el 50% de los defectos, tomando en cuenta la experiencia del programador y la complejidad de las tareas.

IBM Rational Unified Process ® (RUP ®)

Es un marco de trabajo de procesos integral que ofrece a la industria las buenas prácticas en el desarrollo de software, instalación y puesta en funcionamiento, para la gestión eficaz de los proyectos.

RUP se basa en un conjunto de bloques de construcción, o los elementos de contenido, describiendo lo que se va a producir, los conocimientos necesarios y el paso a paso de cómo lograr los objetivos de desarrollo concretos. Los principales "bloques de construcción", son los siguientes:

- Papeles**:** un rol define un conjunto de habilidades relacionadas, las competencias y responsabilidades.
- Productos de trabajo**:** Un producto de trabajo[27] representa algo como resultado de una tarea, incluyendo todos los documentos y modelos producidos durante el trabajo en el proceso.
- Tareas**:** Una tarea describe una unidad de trabajo asignado a una función que proporciona un resultado significativo.

Dentro de cada iteración, las tareas se clasifican en nueve disciplinas: seis "disciplinas de la ingeniería" (Modelamiento de Negocio, Requisitos, Análisis y Diseño, Implementación, Prueba, Instalación) y tres disciplinas de apoyo (Configuración y Gestión del Cambio, Gestión de Proyectos, Medio Ambiente). (IBM, 2010)

Un proyecto manejado con RUP cumple con 4 fases en su ciclo de vida:

[27] En inglés se lo referencia como "WorkItem"

Figure 10 Modelo Iterativo planteado por RUP

Figura 11: Proceso Iterativo de la Metodología RUP

- La fase Inicial[28]: el objetivo es la definición del alcance adecuado para validar costos y presupuestos. Se desarrollan casos de uso, plan de proyectos, medición del riesgo inicial y definiciones generales del proyecto.
- La fase de Elaboración: el objetivo es la mitigación de los riesgos claves para la finalización de la etapa de análisis. Se realiza casos de uso detallados, descripción y ejecución de la arquitectura de software, plan de programación y desarrollo de prototipos.
- La fase de Construcción: el objetivo es la construcción del sistema, con enfoque en sus componentes y características.
- La fase de Transición: el objetivo es el 'tránsito' o traslado de la aplicación desde el ambiente de desarrollo a producción. Se realiza las etapas de entrega y capacitación a los usuarios finales, validación de las expectativas de los usuarios y pruebas de calidad.

[28] A esta fase se la denomina en la literatura en inglés como "Inception Phase".

RUP sugiere la incorporación de 6 buenas prácticas:

- Desarrollo interactivo: Conocer a profundidad los requerimientos
- Manejar los requerimientos: siempre tener en mente los requerimientos de los usuarios.
- Usar Componentes: Dividir el proyecto para que sea posible analizarlo y probarlo por partes.
- Modelos Visuales: Utilizar diagramas visuales para representar las interacciones entre componentes, usuarios, etc. Se recomienda el uso de UML[29].
- Verificar Calidad: Siempre hacer que la ejecución de pruebas sea el mayor enfoque del proyecto en un punto específico.
- Controlar Cambios: Los proyectos son creados por muchos equipos, a veces en diferentes localizaciones. Es necesario establecer un control de cambios y sincronización de versiones y verificarlas constantemente.

ISO / IEC[30] 15504 y 12207

Cada vez más, la calidad del software está tomando mayor importancia en las organizaciones por su influencia en los costes finales y como elemento diferenciador de la competencia y de la imagen frente a sus clientes. En este sentido, ISO/IEC 15504 es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos de las organizaciones en la

[29] UML corresponde a las siglas en inglés (Unified Modeling Language). Es un lenguaje de modelamiento estandarizado que se utiliza normalmente como herramienta en las tareas de ingeniería de software orientada a objetos. En Internet existe amplia documentación sobre este estándar.
[30] Se refiere a las abreviaturas en inglés de (International Organization for Standarization) ISO y (International Electrotechnical Commision) IEC.

Figure 12 Estructura de la certificación ISO / IEC para Procesos de Software

adquisición, desarrollo, evolución y soporte de productos y servicios, e ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software.

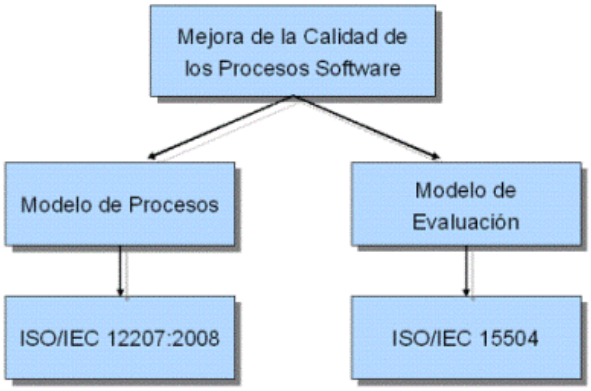


Figura 13: Diagrama de evaluación de la calidad de un Software por ISO

La norma ISO/IEC 15504 establece dos tipos diferentes de evaluaciones para mejorar los procesos de una organización**:** la evaluación por niveles de madurez, donde la organización mejora sus procesos obteniendo una puntuación cuyo alcance es la organización (departamento, proyecto, etc.) y**,** la evaluación por niveles de capacidad, donde la organización obtiene una puntuación a nivel de proceso (gestión de requisitos, planificación de proyectos, etc.). (www.ISO15504.es, 2011)

V-Model

Es un modelo para el desarrollo de sistemas diseñado para simplificar la comprensión de la complejidad asociada al desarrollo de software. En ingeniería del software es usado para definir un procedimiento uniforme para el desarrollo de productos.

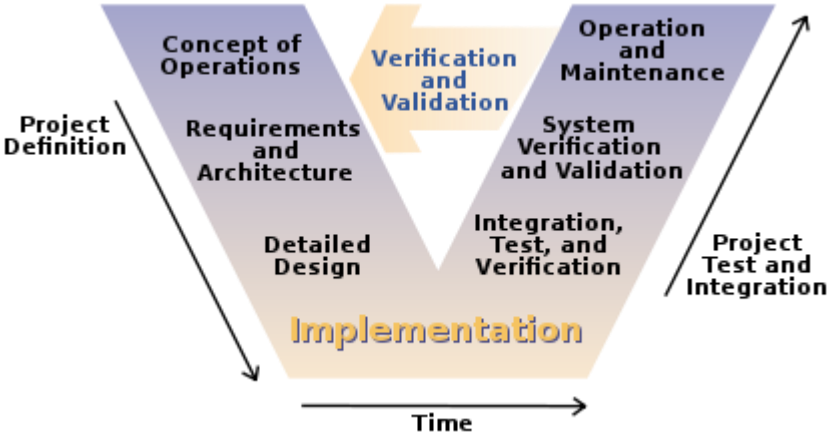


Figura 14: Esquema de la metodología V-Model

El esquema de V-Model tiene la forma de la letra “V”, en la que se representa la secuencia de pasos en el ciclo de vida del desarrollo de software, describiendo los resultados que serán obtenidos durante el proyecto. En el lado izquierdo se encuentra la descomposición de requerimientos, y en el lado derecho se encuentra la integración de las partes.

V-Model provee una guía para la planificación y ejecución de proyectos, buscando alcanzar los siguientes objetivos:

- Minimización de los riesgos del proyecto.

- Mejoramiento y Garantía sobre la calidad.
- Reducción del costo total para el proyecto y para todo el ciclo de vida.
- Mejora en las comunicaciones entre los auspiciantes.

Una de las desventajas de este modelo es que no considera directamente la inclusión de las tareas de operación, mantenimiento y reparación del proyecto desarrollado. (Wikipedia, V-Model, 2011)

Constructive Cost Model (COCOMO)

Es un modelo que permite estimar el costo, el esfuerzo, y el calendario, a la hora de planificar una nueva actividad de desarrollo de software. COCOMO II es la extensión más importante para la COCOMO original (COCOMO 81) modelo publicado en 1981. Se compone de tres submodelos, Cada uno ofrece la mayor fidelidad en la planificación del proyecto y el proceso de diseño. Estos submodelos son la Composición de Aplicaciones, Principios de Diseño, y Post-modelos de la Arquitectura. COCOMO II puede ser utilizado para las siguientes situaciones de decisión importante:

- Durante la planificación de la inversión u otras decisiones financieras relacionadas con un esfuerzo de desarrollo de software,
- Ajuste de los presupuestos de los proyectos y programas, como base para la planificación y el control
- Decisiones sobre las compensaciones o de la negociación entre los costos de software, programación, funcionalidad, rendimiento o calidad de los factores.

- Medición del costo de software y el riesgo de las decisiones de calendario de la gestión.
- Decidir qué partes de un sistema de software deben desarrollarse, la reutilización, el arrendamiento o la compra.
- Establecer estrategias de inversión mixta para mejorar la capacidad del software de la organización, a través de la reutilización, herramientas, madurez de procesos, outsourcing, etc.
- Decidir cómo poner en práctica una estrategia de mejora de procesos.[31]

Existen diversos sitios web en donde es posible realizar los cálculos del modelo COCOMO en línea:

- http://cost.jsc.nasa.gov/COCOMO.html
- http://www.cms4site.ru/utility.php?utility=cocomoii
- http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html
- http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html

Estimación paramétrica

La teoría de la estimación es una rama de la estadística y procesamiento de señales que se ocupa de la estimación de valores basándose en los parámetros obtenidos de una base de datos de empíricos cuando se cuenta con un componente aleatorio. Los parámetros describen un entorno físico subyacente de tal manera que el valor de los parámetros afecta a la distribución de los datos medidos. El algoritmo lo que busca es encontrar valores aproximados para los parámetros desconocidos utilizando las mediciones.

[31] http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html (Consultado en Enero del 2010)

Por ejemplo, se desea estimar la proporción de una población de electores que votarán por un candidato en particular. Esta proporción es el parámetro no observable, la estimación se basa en una pequeña muestra aleatoria de los votantes.

Se utiliza en la estimación del tiempo que tomará el desarrollo de un software específico, utilizando parámetros conocidos y empíricos internos del equipo de trabajo, tal como el tiempo real tomado para el desarrollo de un componentes pequeño, dejando como parámetro no observable el tiempo.

Este modelo matemático tiene su fundamento en conceptos de la estadística, y se aplica a en interpretación científica de experimentos, procesamiento de señales, tratamientos clínicos, control de calidad, encuestas de opinión, telecomunicaciones, manejo de proyectos, determinación de órbitas e ingeniería del software. (Wikipedia, Estimation Theory, 2011)

Modelo Putnam

El modelo Putnam es un modelo de estimación del esfuerzo de desarrollo de software basado en datos empíricos. Dicha teoría fue publicada por Lawrence Putnam en 1978, siendo el pionero en el campo del procesamiento de modelos de software. Como un grupo, los modelos empíricos trabajan recolectando datos de proyectos de desarrollo de software (por ejemplo tamaño y tiempo) y realizando una curva con los datos. Las estimaciones futuras consisten en proveer la variable del tamaño y obtener la variable del tiempo en base a una ecuación que se ajuste a la curva original, claro con errores.

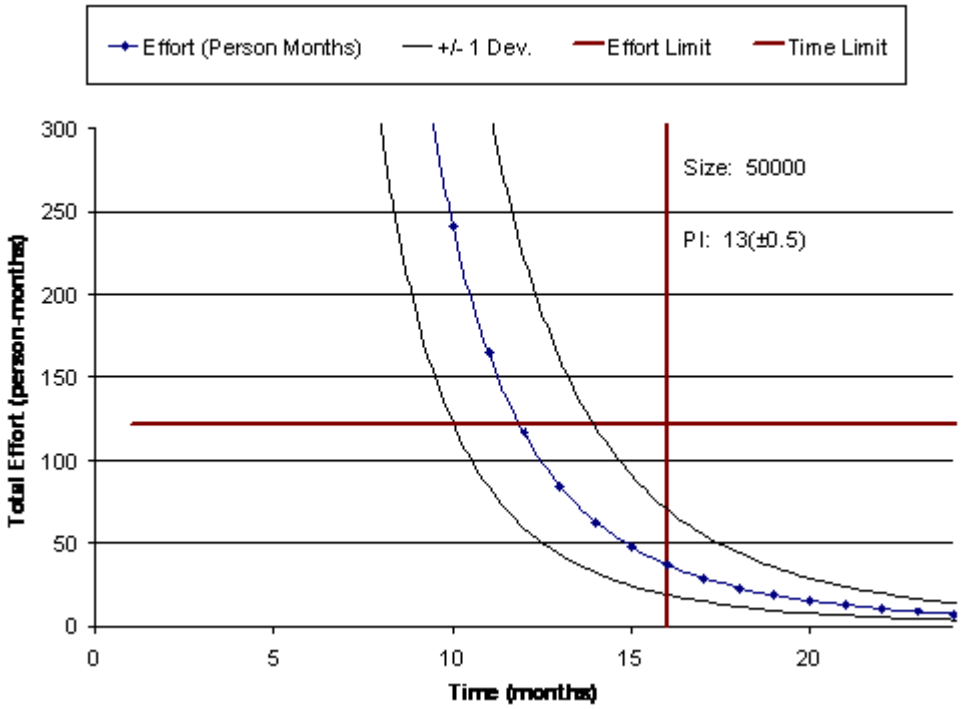


Figura 15 Curva de Esfuerzo en el desarrollo de software del modelo Putnam

Putnam como parte de la investigación realizada en proyectos de investigación y desarrollo de la Armada de los Estados Unidos y General Electrics, encontró que los perfiles del personal que desarrolla software cumplen con la distribución de Rayleigh[32]. Putnam hizo varias observaciones al respecto de los niveles de productividad, logrando obtener la siguiente ecuación:

Equation 1: Fórmula de estimación del esfuerzo de software del modelo de Putnam

$$\frac{B^{1/3} \cdot \text{Size}}{\text{Productivity}} = \text{Effort}^{1/3} \cdot \text{Time}^{4/3}$$

En dónde:

[32] En teoría de probabilidades y estadística, la curva de Rayleigh es una de las más conocidas. Su distribución cumple con una distribución de probabilidad continua. (Wikipedia, Rayleigh Distribution, 2011)

- Size es el tamaño del producto. (cualquier estimador del tamaño es válido como parte de este parámetro, aunque Putnam utilizó siempre las líneas efectivas de código fuente en sus libros)
- B es un factor de escalamiento y es una función del tamaño del proyecto.
- Productivity es la productividad del equipo, que corresponde a la habilidad de una organización de desarrollo de software en producir software de dicho tamaño con un específico margen de error.
- Effort es el esfuerzo total aplicado al proyecto, medido en personas por año.
- Time es el tiempo total en el cronograma del proyecto, expresado en años.

Para obtener una variable específica, es necesario despejar la variable requerida y calcularla en función del resto de variables. Tanto la productividad como el tamaño deben ser basadas en datos empíricos obtenidos en proyectos anteriores, utilizando la misma fórmula pero despejando hacia la variable deseada. (Wikipedia, Putnam Model, 2011)

SEER - SEM

Es una aplicación de software orientado en la gestión de proyectos de software, especializada en estimar, planificar y monitorear los esfuerzos y recursos requeridos para cualquier tipo de proyecto de desarrollo o mantenimiento de software.

La primera versión aplicable del algoritmo fue publicada en 1988 por Dan Galorath y Don Refier, y ha ido mejorando hasta la actualidad, que se encuentra en versión superior a la 7.

SEER es compuesta por un grupo de modelos juntos, que sirven para estimar el esfuerzo, duración, equipo y defectos. Estos modelos responderán a las preguntas de tamaño, tecnología, cálculo de esfuerzo y tiempo, asignación de actividades, cálculo de costos, calculo de defectos, esfuerzo de mantenimiento, entre otros.

La medición del tamaño del software se basa en los modelos de estimación y modelos paramétricos, utilizando métricas que incluyen el uso de líneas efectivas de código fuente[33], puntos funcionales, tamaño basado en funciones y otros. (Wikipedia, SEER - SEM, 2011)

Existen un conjunto de fórmulas que se han utilizado como parte de la estimación del modelo SEER, que se pueden encontrar en los papers públicos de la metodología[34] o la página web oficial del autor Galorath (www.galorath.com). (Galorath, 2011)

Function Point

Es una unidad de medida que sirve para expresar la cantidad de funcionalidad de negocio requerida por un usuario, para ser incorporado en la programación de un sistema de información.

[33] En la literatura se refiere en inglés a la sigla SLOC (source lines of code)
[34] En medio electrónico adjunto al presente documento de Tesis, se encuentran dos artículos referentes a la metodología SEER, llamados "200504-Fischman.pdf" y "SEERforSoftware.pdf".

El costo de cada unidad (en dinero o horas), es calculado a partir de proyectos anteriores, tomando dichos datos para establecer la medición específica. Es un método reconocido como métrica de medición del tamaño del software. (Wikipedia, Function Point, 2011)

La métrica de medición del software por puntos funcionales es una de las más utilizadas en el medio, en vista que se basa en la experiencia del analista para medir cada uno de los puntos a incorporarse en el software. Como parte de esta metodología, no existe una función matemática que permita obtener esta medición de forma exacta.

Proxy Based Estimating

El proceso de planificación de un proyecto requiere una estimación del tamaño del producto de software antes de empezar su desarrollo. Con una estimación del tamaño del software puede obtenerse fácilmente su costo, mediante la aplicación de factores de productividad. Las estimaciones son típicamente requeridas para completar requisitos cuando se conoce muy poco sobre el producto software a construir. La “Estimación Basada en Proxy” produce estimaciones al respecto del tamaño del software, basándose en los desarrollos anteriores para ámbitos de negocio similares.

Un Proxy es una unidad que forma parte del software completo, que puede ser identificado en una etapa temprana del proyecto. Por ejemplo pueden ser el diseño de pantallas, archivos, objetos, entidades lógicas, funciones o puntos funcionales. Los Proxies pueden ser visualizados desde las especificaciones embrionarias de un proyecto, tal como el documento de requerimientos.

Luego, estas pueden ser traducidas a número de líneas de código fuente, utilizando como referencia valores históricos de proxies similares en proyectos pasados. El número de líneas de código predichas, junto con las figuras de productividad del equipo, pueden ser utilizadas para predecir los recursos requeridos para el proyecto.

Un ejemplo de la estimación basada en Proxy puede ser si se decide medir el tamaño de un sistema en función del número de pantallas que serían necesarias para satisfacer un requerimiento. Usando esta medida, es posible establecer el tamaño del software completo basándose en experiencia anteriores, como es el caso de la siguiente tabla, que fue tomada de un equipo de trabajo que utilizaba el lenguaje de programación COBOL (los números representan miles de líneas de código fuente (Chambers & Asociated Pty Ltd, 2006):

Table 1 Tabla de estimación del tamaño de un software basado en la experiencia, para la metodología de estimación PROXY

	Transaction Complexity		
Transaction Class	Low	Medium	High
Create Entity	100	300	1000
Read Entity	50	150	200
Update Entity	200	400	1000
Delete Entity	50	70	150

Program Evaluation and Review Technique - PERT

Es un modelo de manejo de proyectos diseñado para analizar y representar las tareas involucradas en la ejecución de un proyecto. Es comúnmente usado en conjunto con el método de ruta crítica CPM[35].

Este modelo analiza cada una de las tareas involucradas, especialmente al respecto del tiempo que tomará cada una, e identificando a la final el tiempo mínimo en el que es posible realizar todo el proyecto.

Table 2 Tabla de tiempos estimados base para un diagrama PERT

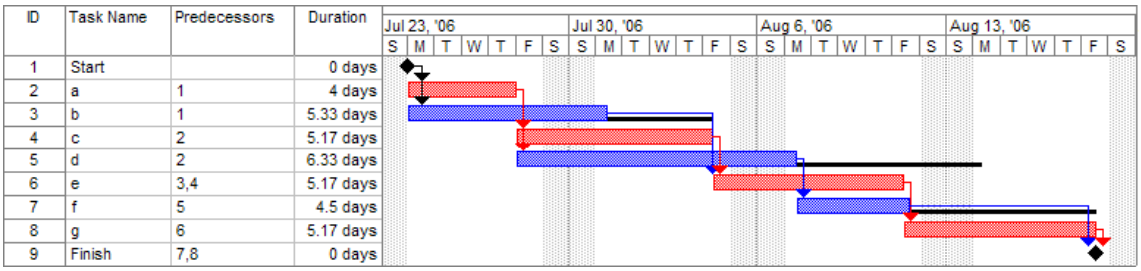
Activity	Predecessor	Time estimates			Expected time
		Opt. (*O*)	Normal (*M*)	Pess. (*P*)	
A	—	2	4	6	4.00
B	—	3	5	9	5.33
C	*A*	4	5	7	5.17
D	*A*	4	6	10	6.33
E	*B*, *C*	4	5	7	5.17
F	*D*	3	4	8	4.50
G	*E*	3	5	8	5.17

Una vez estructuradas las tareas a través de una tabla de estimativos, será posible realizar un diagrama Gantt o un diagrama tipo red, que mejorará la visualización de las secuencias desde las tareas predecesoras (Wikipedia, Program Evaluation and Review Technique, 2011).

En el área del software, es comúnmente utilizada esta técnica de evaluación de ruta crítica para establecer las tareas que deben realizarse con mayor prioridad para lograr alcanzar el objetivo final. Durante el proceso de cotización y evaluación previo al arranque de un proyecto de software, normalmente los

[35] Viene de las siglas en inglés Critical Path Method

clientes solicitan la presentación de un diagrama Gantt para facilitar su visualización de la secuencia y etapas del proyecto.



ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days

Figura 16: Representación de una matriz PERT a través de un diagrama Gantt

La desventaja es que los diagramas PERT parten el preconcepto de que cada tarea debe tener definido específicamente el tiempo que tomarán, pero en el desarrollo de software, este tipo de evaluaciones siempre varían, por lo que la mayoría de planificadores realiza incrementos en los tiempos de cada tarea como margen de riesgo.

PRICE

PRICE Systems es una empresa fundada en 1975, que es conocida por contar con un software que desarrolla una metodología llamada PRICE (Programmed Review of Information for Costing and Evaluation) creada por Frank Freiman. Es el primer software que permite la estimación de costos del software en base a parámetros.

Su enfoque se realizó en función de la estimación de tiempos de desarrollo de hardware y de software, y se presentó el modelo llamado True Planning en el año 2003, en donde cuentan con un modelo paramétrico propietario que utiliza

internamente recolección de datos y regresiones estadísticas para determinar el alcance, costo, esfuerzo y cronograma para proyectos de software. La empresa ha obtenido varias patentes al respecto de sus algoritmos incorporados en el software, teniendo como clientes a agencias espaciales, defensa y gobierno. (Wikipedia, PRICE Systems, 2011)

La actual versión de TruePlanning combina información histórica de más de 11000 proyectos, y se va actualizando constantemente, teniendo las últimas mediciones de la industria, 30 años de investigación al respecto de la medición de proyectos de software y un modelo de auto-calibración de tal manera que la estimación va "aprendiendo" a lo largo del tiempo de su uso. (PRICE Systems, 2011)

Calendarización basado en Evidencias

Esta metodología parte del concepto que a los desarrolladores de software no les gusta hacer cronogramas de ejecución, señalando que “las cosas se harán cuando tengan que hacerse”, dejando al administrador del proyecto fuera de control y los cronogramas desarrollados son archivados y olvidados. El problema es que es necesario conocer el tiempo que tomará cada tarea al inicio del proyecto. Esto ha hecho que nadie crea realmente en el seguimiento de un cronograma.

A lo largo del tiempo, la empresa Fog Creek ha desarrollado la metodología llamada Calendarización basada en evidencias[36] o EBS por sus siglas en inglés. El objetivo es basarse en la evidencia, que consiste obtener datos de tiempos históricos y retroalimentación de los equipos. A diferencia de los métodos de estimación tradicionales, esta estimación no entregará una fecha fija definida de entrega del software o de ciertas etapas, sino que entregará una curva, en las cual se presentará la probabilidad de terminar el proyecto en una fecha especificada.

[36] En inglés: Evidence-Based Scheduling

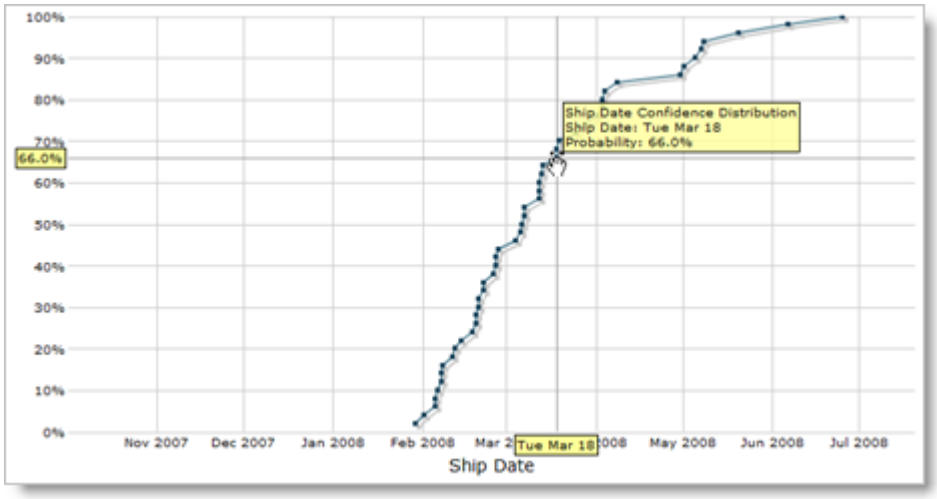


Figura 17: Curva de probabilidad de fechas de término de la metodología de Calendarización basado en Evidencias

Para lograr obtener este tipo de estimación, es necesario cumplir los siguientes pasos:

1. Es necesario dividir un proyecto piloto en sub-tareas, y procurar que ninguna pase de 16 horas de extensión. Esto asegurará que se tenga la mayor cantidad de detalle en la medición.
2. Realizar un estimativo del tiempo de cada tarea e iniciar su desarrollo, y medir los tiempos reales tomados para cada una de las tareas, y realizar un gráfico comparativo.

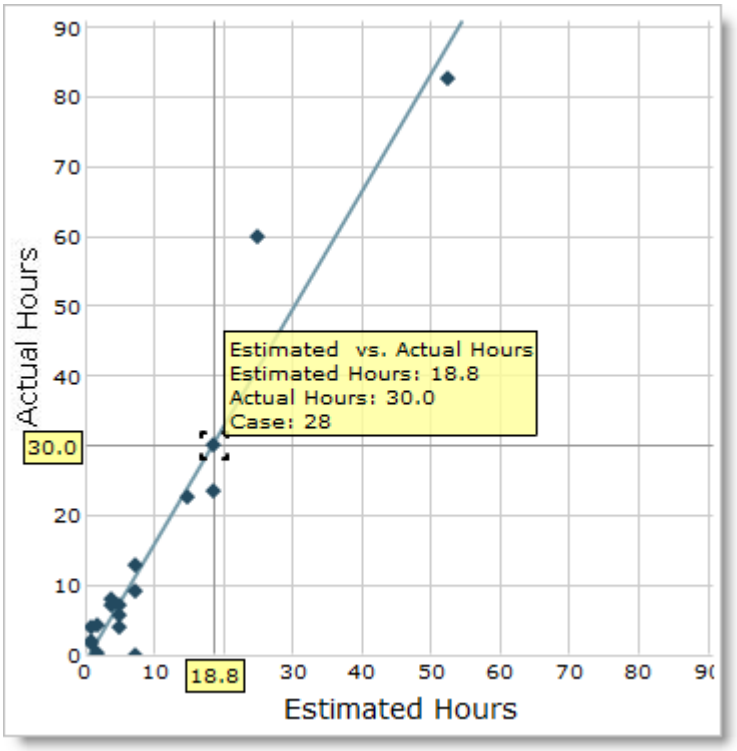


Figura 18: Gráfico Comparativo de medición entre el tiempo real y el estimado.

Luego para cada cifra, se divide el estimado por el actual, obteniendo así la medición de "velocidad". En un ambiente hipotéticamente perfecto, los resultados de la división serían {1,1,1,1,1,1,1..}, o para un ambiente absolutamente imperfecto, los resultados de la división serían {0.1, 0.5, 1.7, 0.2, 0.9…}. En realidad para la mayoría de proyectos, se obtendrá un patrón más o menos definido, en donde se consideran las variaciones normales dentro del desarrollo de un proyecto de software[37], obteniendo resultados como {0.6, 0.6, 0.7, 0.5, 0.6, 0.6, 0.5…}. Esta información nos entrega la información necesaria para que el planificador empiece a ajustar sus prácticas de medición.

[37] Las interrupciones más comunes en el desarrollo de software corresponden a reuniones de planificación, tiempo excesivo en reuniones, distracciones, arreglo de errores, refrigerios y otros.

3. Para simular el futuro, es necesario utilizar el método de Monte Carlo[38] para simular múltiples futuros posibles. La idea sería el crear 100 posibles escenarios, en donde cada uno tiene el 1% de probabilidad de ocurrencia. Si se hace un gráfico con dichas probabilidades, se tendrá una distribución probabilística de los tiempos estimados para el proyecto.
4. Al repetir el algoritmo varias veces, será posible el mejoramiento en las estimaciones, y considerando que su aplicación inclusive considerará las variaciones anómalas sobre el proyecto. (Spolsky, 2007)

Planning Poker

Es una técnica que permite establecer consensos dentro de un equipo de desarrollo de software, al respecto de la estimación de tiempo y esfuerzo requerido para determinadas tareas. Comúnmente es utilizado en conjunto con las metodologías ágiles de desarrollo de software.

La técnica consiste en contar con una baraja de cartas que cumplan una variación de serie de Fibonacci, de la siguiente manera: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, más una carta con un interrogante y una con una taza de café[39].

[38] El método de Monte Carlo es una clase de algoritmo computacional que depende de un muestreo de números aleatorios para calcular sus resultados. Es comúnmente utilizado en simulaciones físicas y matemáticas, y se utiliza cuando no es posible establecer un resultado exacto a través de un algoritmo determinístico. (Wikipedia, Monte Carlo method, 2011)

[39] El ejercicio también puede ser realizado con una baraja normal de cartas, tomando el As, 2, 3, 5, 8 y la K.

Figura 19: Barajas especializadas para Planning Poker

El procedimiento inicia con un moderador que explica las reglas del juego, quien no intervendrá en el mismo. A continuación el responsable de las características a ser implementadas en el software, expone una de ellas al grupo, quienes podrán hacer preguntas sobre el mismo. Cada participante pone una carta en la mesa en donde un menor valor significa que se puede cumplir con mayor rapidez el requerimiento, y una de mayor valor significa que tomará mayor tiempo. Si se pone la carta de mayor valor, significa que el requerimiento es “imposible” en el momento actual del proyecto. Todos viran sus cartas para mostrar al resto sus estimaciones. Se detectan los que pusieron valores muy bajos y muy altos y se les pide que expongan sus argumentos. Nuevamente se les pide que cada uno re-analice su posición y vuelva a poner las cartas sobre la mesa, repitiendo el proceso hasta que el consenso sea logrado y por ende se haya determinado el tamaño del requerimiento.

Para un proyecto completo, se realiza esta tarea para cada uno de los requerimientos a ser incorporados, logrando así una estimación del tiempo total del mismo.

La ventaja de esta técnica consiste en eliminar la barrera de la discusión de tiempos, y establecer un mecanismo de medición común, que permita la reflexión solamente en los puntos discordantes, sin perder de vista la opinión de todos a la vez. (Wikipedia, Planning Poker, 2011)

PARTE II: Ejecución del estudio

Capítulo 4. Ejecución de la encuesta

Diseño de la herramienta de encuestas

El objetivo de la elaboración de una encuesta para este estudio consiste en validar las hipótesis planteadas para esta tesis, especialmente en lo referente a determinar si los atrasos en los proyectos de software en el Ecuador realmente son reales y corresponden a un problema real tangible, o si por el contrario corresponde a una percepción desde un solo punto de vista.

Las condiciones que deben plasmarse durante la encuesta son las siguientes:

- Manejar lenguaje jovial y de camaradería, común entre las comunidades los desarrolladores de software a nivel mundial[40].
- Obtener la mayor información posible de la ”demografía tecnológica”, de tal manera que sea posible determinar si existe algún tipo de relación con respecto a los retrasos en el tiempo del proyecto.
- Enfocar la terminología y las preguntas al ambiente local, es decir, al Ecuador. Sin embargo tener en cuenta también que existe diversidad al interior del país.

[40] La mayoría de blogs y artículos realizados por los desarrolladores de software a nivel mundial, sin importar la tecnología o su origen, no son formales, sino en un lenguaje directo, de compañerismo, de colaboración, y normalmente de forma gratuita. Es por ello que se ha llamado a este tipo de grupos de personas como “Comunidades”.

- En vista que se trata de realizar una encuesta a personas que se encuentran todos los días trabajando con herramientas tecnológicas, es necesario que la encuesta utilice estos medios para llegar más fácilmente.
- Es importante demostrar la formalidad de la encuesta, mostrando los logotipos de la Universidad, nombre del director y nombre del investigador, además incluir datos de contacto. Esto es una condición elemental para que los participantes consideren la encuesta como válida y real. Esto es importante porque actualmente en el Internet circundan muchos tipos de solicitudes de encuestas o similares, que lo que buscan es obtener información de los correos electrónicos para luego hacer envío de correos no deseados (SPAM).

Planteamiento de las Preguntas

Es importante que antes de presentar las preguntas propiamente dichas, se explique los objetivos de la encuesta, pero en lugar de hacer una descripción formal, se utilice un lenguaje directo y jovial, de tal manera que se motive a las personas a continuar con las encuestas.

Se propone presentar el siguiente texto como introducción:

Maestría en Alta Gerencia
Creación de un 'Modelo de Gestión' para empresas de Software en el Ecuador
Dirección de Tesis: Msc. Oswaldo Espinosa
Investigador: Andrés Bastidas Fuertes

Introducción

Lo más probable es que si abriste a esta página es porque estás ligado con el desarrollo de software en el Ecuador....

O tal vez te contaron que se estaba haciendo un estudio interesante al respecto de las personas que trabajamos en desarrollo de software...

O quizás porque alguien te mintió diciéndote que si llenas una encuesta participas en un sorteo de un "auto último modelo"...

y bueno..., lo importante es que estás aquí... ... así que te cuento mi idea antes que quieras irte:

Estoy tratando de resolver un problema que me parece te interesa a ti tanto como a mí, se trata de encontrar los motivos por los cuales las personas que estamos ligadas al desarrollo de software en el Ecuador siempre tenemos

problemas con las fechas de entrega finales, (lance la primera piedra el desarrollador ecuatoriano que no se ha amanecido por un proyecto... [41]).

La hipótesis consiste en buscar los motivos por los cuales nos atrasamos con las fechas de entrega, y cómo podemos ponernos de acuerdo las personas que estamos en la industria ecuatoriana del software para aplicar "buenas prácticas" en la planificación y gerencia de este tipo de proyectos, para ver si es posible obtener los mismos resultados sin tener que hacer tanto sacrificio.. (y poder salir a la misma hora que salen los profesionales de las otras áreas !!!)

Mi nombre es Andrés Bastidas, soy gerente de una de las empresas de software ecuatorianas, y este estudio es parte de una investigación que estoy desarrollando para mi tesis de grado de una Maestría en Gerencia. La encuesta está formada solamente por XX preguntas. Lo que quiero es que me ayudes llenando la encuesta, para poder completar el estudio que vengo realizando en éstos últimos meses.

Por último (veámos si logro convencerte..), recuerda las complicaciones que uno tiene para hacer su tesis en donde es difícil obtener las encuestas llenas para hacer un buen estudio, por eso confío en la colaboración y amabilidad característica de los desarrolladores (o alguien no ha usado un foro para resolver un problema técnico !!), así que... si es que te quedan por ahí unos minutitos libres, ayúdame llenándola por favor, vale la pena !!

[41] Este tipo de aseveraciones son clave para la motivación para el llenado de la encuesta, porque desde que tengo cercanía con los profesionales de sistemas, no he encontrado uno que me diga que nunca se ha amanecido por un proyecto. De hecho esta costumbre viene plasmada desde los mismos inicios de la carrera de sistemas.

Por cualquier duda, comentario, queja, reclamo, felicitación, invitación, chiste, etc.., por favor enviar un correo electrónico a andres@smartwork.com.ec

Para finalizar con la introducción, he considerado la posibilidad de registrar los datos de contacto de los que hayan participado en la encuesta. Esto porque como uno de los objetivos al finalizar el estudio, es la publicación y distribución del documento a la mayoría de personas que sea posible, logrando que la aplicación de las buenas prácticas se generalice en las diversas empresas y profesionales del área.

Pregunta 1:

El objetivo de la primera pregunta consiste en determinar si es que existe más retrasos en una tecnología u otra. La mayoría de desarrolladores hemos programado en varios lenguajes como parte del proceso de aprendizaje y exploración, pero regularmente se utiliza un conjunto de herramientas para el trabajo profesional, por lo que es importante señalar que se trate de los últimos años solamente.

Se propone presentar el texto de la primera pregunta de la siguiente manera:

Tomando en cuenta los proyectos de software en los que has participado en los últimos dos años, Qué herramientas tecnológicas has utilizado principalmente?

Lenguaje de Programación	
.Net Framework	Java
PHP	JSP
Silverlight	Flash
Visual Basic	C / C++
HTML / CSS / Javascript	Cobol
Basic	Python
Delphi	Power Builder
Otro	

Plataforma	
Windows	Linux
Unix	Web
Mainframe	Macintosh
Mobile	Otro

Base de Datos	
SQL Server	Oracle
MySQL	Postgres
DB2	Sybase
Interbase	FoxPro
Otro	

Cómo vendo mi software?	
Software a Medida	Software Masivo / en Caja
Consultoría de Software	Implementación de soluciones
Venta de licenciamiento	Offsourcing / Offshoring

Otro	

Orientación	
Software Libre	Software Propietario
Otro	

Mercado vertical	
Salud, Hospitales, Laboratorios Clínicos	Educación, Colegios, Universidades
Producción Industrial, Maquinarias	Telecomunicaciones, Proveedores de Internet
Banca, Cooperativas, Seguros	Comercio, Bodegas, Importaciones
Transporte, Logística	Seguridad, Vigilancia
Otro	

Pregunta 2:

El objetivo de la segunda pregunta consiste en determinar el rol que cumple el encuestado dentro de los ciclos de desarrollo de software[42], y más que nada determinar si es que es posible existe una relación directa o inversa entre los retrasos del software y el nivel de experiencia dentro de este tipo de negocios.

Se propone presentar el texto de la segunda pregunta de la siguiente manera:

[42] Como hemos visto en la sección de Estado del Arte, prácticamente todas las metodologías revisadas cumplen con un ciclo de Visionar, Analizar, Programar, Probar, Instalar. Unas con ciclos completos y otros con ciclos cortos, pero siempre cumpliendo ese esquema.

A continuación se presentan los diversos roles que una persona juega dentro de un proyecto de software. Por favor selecciona en qué nivel de experiencia te ubicas para cada una de las opciones presentadas:

	Tengo mucha experiencia	**Tengo cierta experiencia**	**Estoy recién experimentando**	**Conozco, pero no he aplicado**	**Nunca he tenido la oportunidad**
Gerente de una empresa de software					
Gerente de proyectos de software					
Arquitecto					
Jefe de Equipo de Programación					
Desarrollador / Programador					
QA / Tester					
Soporte Técnico a Usuarios					
Implementador					

El usuario deberá seleccionar, para cada uno de los perfiles presentados, una de las opciones presentadas en la cabecera.

Pregunta 3:

El objetivo de la tercera pregunta consiste en realizar una validación al respecto de si la percepción de que la mayoría de proyectos de software en el Ecuador tienen problemas con el cumplimiento de tiempos es real o solamente se trata de un punto de vista específico.

Se propone presentar el texto de la tercera pregunta de la siguiente manera: Tomando en cuenta los proyectos en los que has estado involucrado, los proyectos que has visto desarrollarse por otras personas y en general todos los proyectos que has escuchado realizarse por profesionales ecuatorianos:

Qué porcentaje de proyectos crees se han atrasado con sus fechas de entrega?	(El usuario ingresa el porcentaje sobre el 100%) [43]

Pregunta 4:

Como resultado de conversaciones directas realizadas previo a la elaboración de la encuesta, se pudo evidenciar que los gerentes y planificadores del desarrollo de software tienen diferentes posiciones al momento de otorgar la prioridad a las tareas y la consecuente distribución del tiempo.

En las diferentes metodologías de desarrollo de software estudiadas en la sección de Estado del Arte, se puede ver que también existen diversas perspectivas en la distribución del tiempo a lo largo de las etapas, pues en

[43] En este caso no se ha considerado establecer una escala tal como "Todos, la mayoría, varios, pocos, etc", por razón que será más útil para el análisis posterior el valor numérico definido, y así tener un mayor rango de posibilidades de respuesta. Esta consideración se ha tomado por tratarse de una pregunta central del estudio.

metodologías como RUP o MSF se da mucho énfasis a las etapas de análisis, mientras que en las metodologías ágiles, tales como SCRUM o Extreme Programming, se da mayor importancia a la programación y el código fuente en sí.

Hace cuatro años se tuvo la oportunidad de asistir a una charla dada por Jorge Oblitas, que en ese tiempo cumplía el cargo de apoyo a las comunidades de profesionales de sistemas que utilizaban la tecnología Microsoft en la región Andina. Durante su charla exponía algunas estadísticas en las que demostraba que durante los procesos de desarrollo de software se tienen una alta cantidad de reprocesos y retrabajos por culpa de no elaborar un estudio lo suficientemente amplio como para determinar exactamente cada una de las características del software. En la conclusión de su charla, él recomendaba que es muy importante asignar más tiempo a la etapa de análisis que a la etapa de programación, para evitar reprocesos y por ende evitar retrasos en la entrega de los productos. Por este motivo yo acotaba en la etapa de preguntas que a la larga lo que esperan los usuarios finales es el software funcionando, y no solamente un conjunto de documentos que describen lo que debería tener el software, pues señalaba que para lograr hacer un documento así de completo, sería necesario llevar a cabo el software como tal, para ver qué tipo de aspectos y condiciones podrían variar, ocasionando más bien un retraso mayor. Independientemente del resultado de la discusión formada gracias a esta charla y de otras experiencias similares[44], es evidente que hay puntos de

[44] En diversas experiencias que durante mi vida profesional he desarrollado, he visto que no hay un estándar que nos indique cómo distribuir el esfuerzo en el desarrollo del software a lo largo de sus etapas, pues es claro que no existe el proyecto de software "perfecto", ya que no habría un auspiciante que

vista contrapuestos y dispares, que nos orientan a pensar que no hay una teoría que indique cuál es la distribución óptima del tiempo en las etapas, y pienso que sería muy difícil unificar este criterio.

Para tratar determinar si es que las personas que asignaron mayores esfuerzos a una etapa que a otra generaron mayores retrasos, planteo la cuarta pregunta de la siguiente manera:

Selecciona 3 proyectos que a tu parecer son los más representativos de tu carrera. Ingresa en las casillas las formas cómo has distribuido realmente el tiempo para estos proyectos:

	Definiciones Iniciales	Análisis y Diseño	Programación	Estabilización, Pruebas, Ajustes	Instalación, Capacitación, Salida a Producción	Soporte
Proyecto 1	%	%	%	%	%	%
Proyecto 2	%	%	%	%	%	%
Proyecto 3	%	%	%	%	%	%

Pregunta 5:

A pesar que la distribución del tiempo es importante para determinar las posturas tomadas por los planificadores, esto no nos indica a ciencia cierta qué tan amplio fue el retraso y cuáles fueron los motivos del retraso.

permita desarrollar todas las tareas con alta profundidad, y si así fuera, el tiempo no tendría que ser una limitante. Otro ejemplo de esto fue en una institución financiera que participé en unos proyectos de software, en dónde su énfasis era claramente orientado hacia las pruebas, y específicamente sobre las pruebas de rendimiento y carga, ya que ellos asignaban mucho tiempo del desarrollo para tratar este punto específico, a veces más que la misma etapa de programación.

Esta pregunta se ha planteado en el sentido de determinar la magnitud del retraso y cuáles son los problemas más comunes que son los orígenes del problema.

Se propone realizar el texto de la quinta pregunta de la siguiente manera:

Para los mismos 3 proyectos elegidos en la pregunta anterior, realice la comparación entre los tiempos planificados originalmente y los tiempos realmente consumidos.

	Tiempo planificado (meses)	Tiempo Realmente consumido	Si hubo retraso, cuál fue el principal motivo?
Proyecto 1	#	#	(Selección de opciones)
Proyecto 2	#	#	(Selección de opciones)
Proyecto 3	#	#	(Selección de opciones)

Las opciones presentadas en la última columna serían las siguientes:

- El proyecto no se atrasó
- Hubo cambios en la estructura del equipo de programación
- Problemas con la arquitectura o el diseño técnico
- Desconocimiento del negocio
- Problemas ocasionados por el usuario final en la entrega
- Había problemas internas constantemente, hubo desmotivación
- Errores en la planificación del tiempo
- La tecnología utilizada no se conocía bien

- Sobrevaloración de las capacidades del equipo
- El equipo de trabajo era muy pequeño, se necesitaba más gente
- Hubieron demasiados bugs[45] que corregir
- Errores en la definición del alcance del desarrollo

Pregunta 6:

Otro punto de vista dividido entre los planificadores de software es al respecto de la concepción de si es que el software puede o no puede ser "predicho" a través de un algoritmo matemático o metodología específica de forma exacta.

Esta pregunta va orientada a determinar si las concepciones del mercado nacional van orientadas hacia la predictibilidad, o si más bien van orientadas a la gestión del riesgo, y en base a ello determinar los niveles de atraso ocasionados por ambos puntos de vista.

Se propone realizar el texto de la sexta pregunta de la siguiente manera:

El desarrollo de software normalmente implica la intervención de modelos y herramientas complejas para su ejecución. Puede decirse entonces que para la persona que planifica este tipo de proyectos se torna difícil encontrar con exactitud el tiempo que tomará el desarrollo de software. Cree usted que:

La medición del tiempo que tomará desarrollar un proyecto de software es "predecible" exactamente, a través de fórmulas, métodos matemáticos o secuenciación de tareas con el uso de una herramienta de formulación de	SI / NO

[45] Un bug en el argot del desarrollo de software se refiere a un problema de programación encontrado en la programación de una herramienta, y que se presenta solamente cuando el software se encuentra ya en ejecución. En la encuesta se ha utilizado este término por razón que se estima que el target objetivo de la encuesta sean desarrolladores de software o disciplinas similares.

proyectos.	
La medición del tiempo que tomará desarrollar un proyecto de software “no es predecible”. El software tiene demasiadas variaciones en el diseño y la programación, que se torna muy complicado determinar exactamente el tiempo que tomará desarrollarlo.	SI / NO
El software no es predecible de forma exacta, pero con el uso de las fórmulas matemáticas se puede llegar a una aproximación, y luego se debería hacer un análisis del escenario, viendo qué tipo de cliente, qué tipo de producto y qué tipo de riesgos están involucrados, para así ajustar la predicción del tiempo que tomará.	SI / NO
Otra	Texto

Pregunta 7:

En la sección del marco teórico se realizó la explicación de los diversos puntos de vista que pueden tomar los planificadores de software al respecto de la arquitectura de software. Esta pregunta va orientada en determinar qué nivel de conciencia existe en el mercado ecuatoriano al respecto de la teoría de arquitectura de software y cuáles son sus formas de aplicación.

Planteo el texto de la séptima pregunta de la siguiente manera:

La arquitectura de software es una tarea técnica realizada por un especialista con mucha experiencia en desarrollo de software. Su tarea es entender las funcionalidades que deberá implementar el sistema, estudiar el escenario y

plantear una estructura tecnológica que facilite la etapa de programación para el equipo de desarrollo. Cree usted qué:

La arquitectura de software es 'trascendental' para el éxito del proyecto, ya que en ella se determinarán todos los lineamientos tecnológicos antes de iniciar el proyecto.	SI / NO
La arquitectura de software 'no es trascendental' para el éxito del proyecto, pero se realiza cierto tipo de estudio parecido para cumplir con los requisitos de documentación.	SI / NO
Normalmente 'no realizamos un estudio de arquitectura de software', por no considerarla útil, o porque implica costo y tiempo que no justifica invertir. Se usan 'arquitecturas' que ya se encuentran definidas en el Internet, o en proyectos ya desarrollados anteriormente.	SI / NO
No conozco qué es o para qué sirve la arquitectura del software	SI / NO
Tengo otro concepto sobre la arquitectura de software	Texto

Pregunta 8:

Una vez que hemos determinado con las preguntas anteriores los lineamientos conceptuales y las prácticas generales utilizadas para cada uno de los encuestados, se plantea con esta pregunta profundizar específicamente en el problema de la medición de tiempos, buscando principalmente encontrar cuál de los métodos matemáticos o no matemáticos son los más utilizados en el medio ecuatoriano.

Se plantea la presentación de la octava pregunta de la siguiente manera:

Imaginemos que usted es la persona que realiza las cotizaciones para la programación de un nuevo software. Un día se acerca un cliente hacia donde usted y le expone durante 4 horas las ideas que él tiene para el desarrollo de un nuevo sistema, usted obtiene toda la información que pudo preguntarle durante ese lapso de tiempo. De qué manera colocaría la medición del tiempo que se colocará en la propuesta:

Modelo COCOMO	SI / NO
Estimación Paramétrica	SI / NO
Modelo Putnam	SI / NO
SEEM SER	SI / NO
Análisis de Puntos Funcionales	SI / NO
Proxy Based Estimation (PROBE)	SI / NO
PERT	SI / NO
PRICE	SI / NO
Evidence Based Scheduling	SI / NO
Planning Poker	SI / NO
Me basaría en mi experiencia, analizando uno por uno los requerimientos que se implementarán. Al final podría un margen de tiempo adicional como 'colchón' de tiempo.	SI / NO
Me basaría en mi experiencia, analizando al proyecto entero, y comparándolo con proyectos desarrollados anteriormente y proyectos	SI / NO

realizados por otras empresas. En base a esto se propondría un tiempo razonable en función de competir con el mercado.	
Haría una reunión con el equipo de desarrollo, les plantearía los requerimientos solicitados por el cliente, y les pediría que se mida el tiempo de cada una de las tareas detectadas. Al final pondría un margen de tiempo adicional como colchón.	SI / NO
Yo realizaría la medición de la siguiente manera:	Texto

Pregunta 9:

De la misma manera que la pregunta anterior, el objetivo de esta pregunta es la de profundizar específicamente sobre el problema de la estimación de presupuestos para los proyectos de software, y determinar si ellos están ligados directamente con respecto a su elección en la medición del tiempo, o más bien están ligados a variables externas, como variables de mercado o de capacidad del cliente.

Se considera presentar la novena pregunta de la siguiente manera:

Al igual que la pregunta anterior, imaginemos que usted es la persona que realiza las cotizaciones para la programación de un software. En base al estudio elegido previamente, usted ya conoce cuánto tiempo se estima durará el desarrollo del proyecto de software. De qué manera se colocaría el precio del software en su cotización?

Utilizaría uno de los métodos matemáticos disponibles para valoración de	SI / NO

costos (COCOMO, SEEM-SER, PERT, PROBE, etc) para determinar los costos del proyecto. Al valor resultante agregaría un margen de utilidad y obtendría el precio.	
Tomaría el tiempo total calculado, lo dividiría para el número de programadores disponibles, en función de esto sabría el tiempo de dedicación que tendrían cada uno de ellos en todo el proyecto, a esto lo multiplicaría por el sueldo mensual de ellos. A esta suma le pondría los gastos administrativos y aplicaría un margen de utilidad. Con esto tendría el precio para cobrarle al cliente.	SI / NO
Haría el mismo proceso del punto anterior, pero lo ajustaría hacia arriba o hacia abajo en función de los precios del mercado que ofrecen las otras empresas de desarrollo, ya que con ello podría ganar utilidades adicionales o disminuir el precio para poder competir.	SI / NO
Analizaría la capacidad financiera del cliente y el beneficio proyectado que él obtendrá con el software. En función de eso establecería un precio acorde con su capacidad económica y la recuperación de su inversión.	SI / NO
Me basaría en mi experiencia de proyectos anteriores exitosos, para establecer una proporcionalidad que me permita establecer el tamaño del proyecto actual, y por ende su precio.	SI / NO
Yo realizaría la medición de otra forma	Texto

Pregunta 10:

Como parte de la encuesta, es importante analizar adicionalmente otras variables que no necesariamente son responsabilidad del planificador inicial del

software, sino más bien del jefe de equipo o gerente de proyecto, orientado principalmente determinar las formas adoptadas en las relaciones internas en el área de producción de este tipo de negocios, el equipo técnico, los programadores, diseñadores, analistas, etc.

El objetivo de esta pregunta es la de averiguar si es que el factor del clima laboral y contar con un ambiente adecuado de trabajo influye en los desfases de tiempo y presupuesto de los proyectos.

A continuación se presenta el planteamiento de redacción para la décima pregunta:

Qué tanto influye el contar un buen ambiente de trabajo (llevarse bien

con los compañeros, apertura al diálogo, colaboración interna del equipo,

etc.), para lograr los objetivos del desarrollo de software?

Es muy importante e indispensable	SI / NO
Es importante, pero no indispensable	SI / NO
Uno no siempre puede elegir con quién va a trabajar, así que hay que adaptarse al entorno, pero si no se puede, depende del profesionalismo de cada persona.	SI / NO
No es tan importante, a la larga es responsabilidad de cada persona cumplir con sus tareas independientemente de su entorno.	SI / NO
No es importante, tanto así que sería posible trabajar de forma remota por medio de Internet sin mucha interacción con los compañeros e igual	SI / NO

se obtendrían los mismos resultados.	

Pregunta 11:

Uno de los problemas que a percepción personal se considera es uno de los más impactantes en la generación de retrasos es el hecho de que una persona en la mitad del tiempo del proyecto se retire del equipo, pues la acumulación de conocimientos en el desarrollo de software no es tan fácil de transmitir.

Es por ello que el objetivo de esta pregunta es la de validar esta posición y determinar si en la industria ecuatoriana de software, este problema es visualizado como “importante” o si es considerado como una acotación irreal.

Planteo la presentación de la onceava pregunta de la siguiente manera:

Cuando una persona de un equipo de desarrollo se retira en el transcurso de un proyecto de software, generalmente ocasiona un retraso en el cronograma por motivo de los conocimientos adquiridos en el proyecto. ¿En los proyectos que usted ha participado, cómo se ha mitigado este problema?

Lo mejor es contar con un gerente de proyecto, que es el encargado de conocer todos los pormenores de las tareas de su equipo de trabajo. Cuando una persona se va, el líder fácilmente podrá capacitar a otra persona para que continúe con el proceso.	SI / NO
Para eso es necesario documentar cada punto desarrollado, así no será difícil que una nueva persona entienda lo que se venía haciendo a partir de la documentación.	SI / NO

Se organizan los equipos de programación, de tal manera que cada persona conozca sus tareas y las tareas de otra persona. Si una persona se retira, al menos habrá una persona que conoce lo que se estaba haciendo.	SI / NO
Se utiliza “Extreme Programming” u otro similar, de tal manera que se tengan dos personas trabajando sobre el mismo código fuente. Si se va una persona, la otra persona tendrá todo el conocimiento para continuar con el proyecto, sin parar ni un minuto del desarrollo en nuevas capacitaciones.	SI / NO
No se ha tratado a éste como un problema crítico. Cuando se da este caso, se analiza las estrategias a seguir en el momento que se den.	SI / NO
Esto no es un problema, simplemente la persona que sale, debe capacitar a una persona que asuma el cargo.	SI / NO
Yo lo realizaría de la siguiente manera:	Texto

Pregunta 12:

Uno de los problemas que implica mayor conflicto en la gerencia de proyectos de software es la administración de requerimientos del usuario, esto es dado porque el software mantiene un proceso de mejora constante y mientras se va diseñando y programando, los usuarios van clarificando sus ideas y van modificando sus puntos de vista al respecto de ciertos conceptos aplicados para los sistemas.

El gerente del proyecto debe dilucidar cuáles cambios a los requerimientos son "asumibles" dentro del proyecto y cuáles deben considerarse para siguientes versiones del desarrollo. Los clientes normalmente presionarán para contar dentro de la misma versión las últimas mejoras propuestas, pero en caso de aceptar su inclusión, necesariamente implicará una modificación de los cronogramas y en algunos casos se asumirá implícitamente un retraso.

La negociación con el cliente es constante al respecto de los requerimientos, tiempo y recursos. Las empresas desarrolladoras que tienen "escritos en piedra" los requerimientos y son rígidos al respecto de las modificaciones, obtienen la percepción de los clientes de no satisfacer sus necesidades, pero por el contrario las empresas desarrolladoras que son muy flexibles en la aceptación de cambios, rara vez cumplen una fecha de entrega e inclusive puede perderse el rumbo general del proyecto. Se busca entonces lograr un equilibrio entre ambas posiciones para llegar a un final del proyecto satisfactorio para el cliente y el equipo de desarrollo.

El objetivo de esta pregunta es la determinación de los métodos utilizados en las empresas ecuatorianas para lograr este equilibrio.

Se propone la presentación de la pregunta Nro. 12 de la siguiente manera:

Los requerimientos dados al inicio de un proyecto por parte de un usuario normalmente no son los que terminan siendo implementados en el software, ya que mientras se va avanzando en el desarrollo se van encontrando nuevas variables, nuevas condiciones, nuevos conocimientos o nuevos escenarios que no se vieron durante la etapa de análisis, y que alteran la planificación del tiempo de ejecución. Esto es un motivo que puede ocasionar importantes

retrasos en los proyectos de desarrollo de software. De los proyectos en los que usted ha participado, cómo se ha tratado de mitigar este problema?

Esto no nos pasa nunca, solamente es necesario establecer documentación formal que cuente con las firmas del cliente, de tal manera que se pueda demostrarle al cliente que ha existido una variación en el diseño, y por ende renegociar los tiempos.	SI / NO
Consideramos que el cliente nos ha contratado para brindarle una solución completa, por ende se le apoyará al cliente en el ajuste de las herramientas ofrecidas hasta llegar a solucionar su problema de negocio, incluyendo los nuevos conocimientos que aparezcan en el camino. En base a esto se negociará con el cliente extensiones del tiempo previsto, demostrándole que los tiempos adicionales serán utilizados para el tratamiento de casos especializados a su negocio.	SI / NO
Es un problema que no debería tener consecuencias de retraso sobre el tiempo estimado originalmente, ya que normalmente planificamos un margen de tiempo adicional que nos permita tratar este tipo de problemas sin salirnos del cronograma.	SI / NO
Nuestra metodología de levantamiento de información y documentación implica que se tengan absolutamente todos los conocimientos del proyecto, antes de empezar la programación. Por ende estos cambios no serán posibles sino hasta que se contrate el desarrollo de la siguiente versión.	SI / NO
Estas variaciones de requerimientos siempre se dan y no son fáciles de predecir. Actualmente tenemos problemas en controlar este problema.	SI / NO

Yo lo realizaría de otra manera	Texto

Pregunta 13:

Mientras participaba de un proyecto de desarrollo de software en donde se cometió varios errores en la planificación de tiempos, lo que nos obligó a dedicar mucho esfuerzo extra, se pudo notar que habían personas que tenían mayor facilidad de asumir estos retos “extremos” sin dejar de ser productivos y otras personas que definitivamente les costaba demasiado hacerlo y disminuyendo su nivel de productividad.

Luego de participar en proyectos de software intensos, se ha visto que la razón de estas diferencias entre los equipos de programación radica inicialmente en la motivación y sentimiento de superación asumida por cada una de las personas y con su nivel de socialización interpersonal, evocando estabilidad emocional y profesional. Por salud mental y para evitar el desgaste emocional, obviamente no es sano estar involucrado siempre en proyectos extremos, pero si este compromiso es cultivado dentro de los proyectos, es posible no solamente evitar atrasos, sino que realizar entregas antes de los tiempos pactados y evitar en todo momento la inversión de esfuerzos adicionales.

Es por ello que el objetivo de esta pregunta consiste en determinar para el mercado ecuatoriano cuáles son los métodos de motivación más utilizados para lograr el comprometimiento con la empresa y los proyectos.

Se propone la presentación de la pregunta Nro. 13 de la siguiente manera:

El trabajo en programación implica un esfuerzo intelectual bastante alto, que en un proyecto de larga duración puede implicar que los desarrolladores acumulen cansancio, disminuyan su ritmo de trabajo, e inclusive lleguen a una desmotivación (muchas veces silenciosa). Cuál crees que es la mejor forma de mantener motivado al equipo de desarrollo?

La mejor forma de motivar al personal es de forma económica. Una persona que se sienta bien remunerada, logrará superar éste tipo de problemas motivacionales.	SI / NO
Los desarrolladores son personas que se encuentran en constante aprendizaje. Una buena forma de motivarles es a través de un constante plan de capacitación sobre la tecnología, que les permita día a día ir creciendo como profesionales.	SI / NO
A través de eventos, paseos, celebración de cumpleaños, deporte, y otro tipo de actividades sociales, que les permita liberar la carga emocional que trae el trabajo.	SI / NO
Colocando en la oficina un Nintendo Wii, un PlayStation, un futbolín u otros medios de distracción temporal, que les permita ir dosificando la carga que trae el trabajo.	SI / NO
El equipo de desarrollo tiene la misma carga de trabajo que los profesionales de otras áreas, y no requiere un plan de motivación específico para ellos. Lo importante es su profesionalismo para lograr los objetivos planteados.	SI / NO
Yo lo realizaría de otra manera	Texto

Pregunta 14:

Una de las variables que se desea incorporar como parte de la encuesta es la que se denominada Variables Culturales, que no tiene mucho que ver con el aspecto técnico del desarrollo de software en sí, sino más bien tiene como finalidad el determinar si las condiciones exógenas y de las costumbres de los usuarios influyen en las decisiones tomadas durante el desarrollo de un proyecto realizado en el Ecuador.

El motivo de incluir esta pregunta nace del hecho que hace algunos años hubo la oportunidad de compartir experiencias con personas extranjeras que desarrollan software, y que posteriormente desarrollaron proyectos dentro del Ecuador, encontrando muchas diferencias y dificultades únicas en nuestro entorno. Una condición que consideraban como más relevante, era el hecho de que los usuarios ecuatorianos son personas que no siguen los procedimientos, siempre buscan un camino alterno, haciendo que los programadores tengan que realizar controles muy finos para evitar problemas en el software, haciendo que a la larga los sistemas vayan obteniendo un nivel de calidad alto.

Por este motivo se ha planteado la pregunta Nro. 14 de la siguiente manera:

Se dice que los desarrolladores de software ecuatorianos somos únicos en el mundo, ya que enfrentamos todos los días a usuarios 'ecuatorianos' que en general no leen los manuales de operación, que no siguen las instrucciones dadas en las capacitaciones, que buscan constantemente los 'caminos alternos' a los procedimientos establecidos en el software.

Esto, sumado a una regulación tributaria, legal y financiera muy variable e impredecible (casos como el cambio de moneda, anexos del SRI, requerimientos del Seguro Social, Ministerio de Trabajo, Superintendencias, etc)

El resultado es el uso 'muy irregular' del software desarrollado, obligando a que se tenga que soportar un gran conjunto de requerimientos para 'sobrevivir' bajo estos escenarios.

Esto hace que los programadores siempre estén pendientes de controlar hasta los caminos más absurdos en su programación a fin de evitar descontroles, e incrementen los niveles de validación, configurabilidad y generalidad de las herramientas. El resultado es que nuestro software tiene alta calidad a los ojos del mercado internacional.

Qué implicación ha tenido esto en su empresa?

Ninguna, es la primera vez que escucho esto.	SI / NO
Los usuarios ecuatorianos siempre le buscan 'la quinta pata al gato', por ende nuestro equipo de QA[46] tiene que realizar pruebas muy exhaustivas para determinar que el software funcionará bien en manos de los usuarios finales. Aún así, los usuarios siempre tienen 'nuevas ocurrencias'.	SI / NO
Lo había escuchado, pero no he podido comparar nuestra programación con la que se hace en otros países.	SI / NO

[46] En el argot del desarrollo de software se utiliza frecuentemente las siglas QA (Quality Assurance), para referirse al equipo o personas que hacen las pruebas del software previo a entregarles a los usuarios finales, asegurándose que se publiquen con la menor cantidad posible de errores de programación.

Esta es una consideración es muy importante, porque llegar a este nivel de detalle en la programación conlleva al consumo de mayor tiempo en desarrollo y mayor tiempo en la estabilización de los productos. Esto debería ser considerado también durante la planificación del tiempo del proyecto.	SI / NO
Lo he escuchado y esto es un riesgo muy alto para la planificación de un software, ya que introduce una incertidumbre muy alta en el momento de validar las entregas de los proyectos, porque los usuarios tienen siempre 'mejores formas' de entender las herramientas.	SI / NO
Mi percepción sobre este problema es diferente	Texto

Pregunta 15:

En el planteamiento metodológico inicial de la presente investigación, se había planteado la realización de entrevistas a profundidad con representantes de diversos sectores de la industria. Sin embargo, al visualizar que ellos también podían proporcionar información importante en el llenado de las encuestas, y los otros tipos de encuestados podían también responder preguntas abiertas que permitan ampliar el espectro de análisis a profundidad, se decidió realizar un cambio metodológico para incorporar en esta pregunta el mismo enfoque que se había considerado para las entrevistas a profundidad.

El objetivo final de esta pregunta es la de otorgarle al encuestado la posibilidad de expresar abiertamente sus opiniones al respecto de los puntos centrales de esta investigación.

Se ha planteado la presentación de la pregunta Nro. 15, de la siguiente manera:

Para concluir con la encuesta, por favor contesta las siguientes 4 preguntas:

1. Qué se debería hacer para evitar retrasos en los proyectos de software?
2. Para usted, cuál es el mejor modelo de gestión de una empresa de software / un equipo de software?
3. Cuál es la mejor manera de hacer rentable / viable el desarrollo de software?
4. Cómo hacer que el software ecuatoriano sea reconocido mundialmente?

Programación del sistema de encuestas

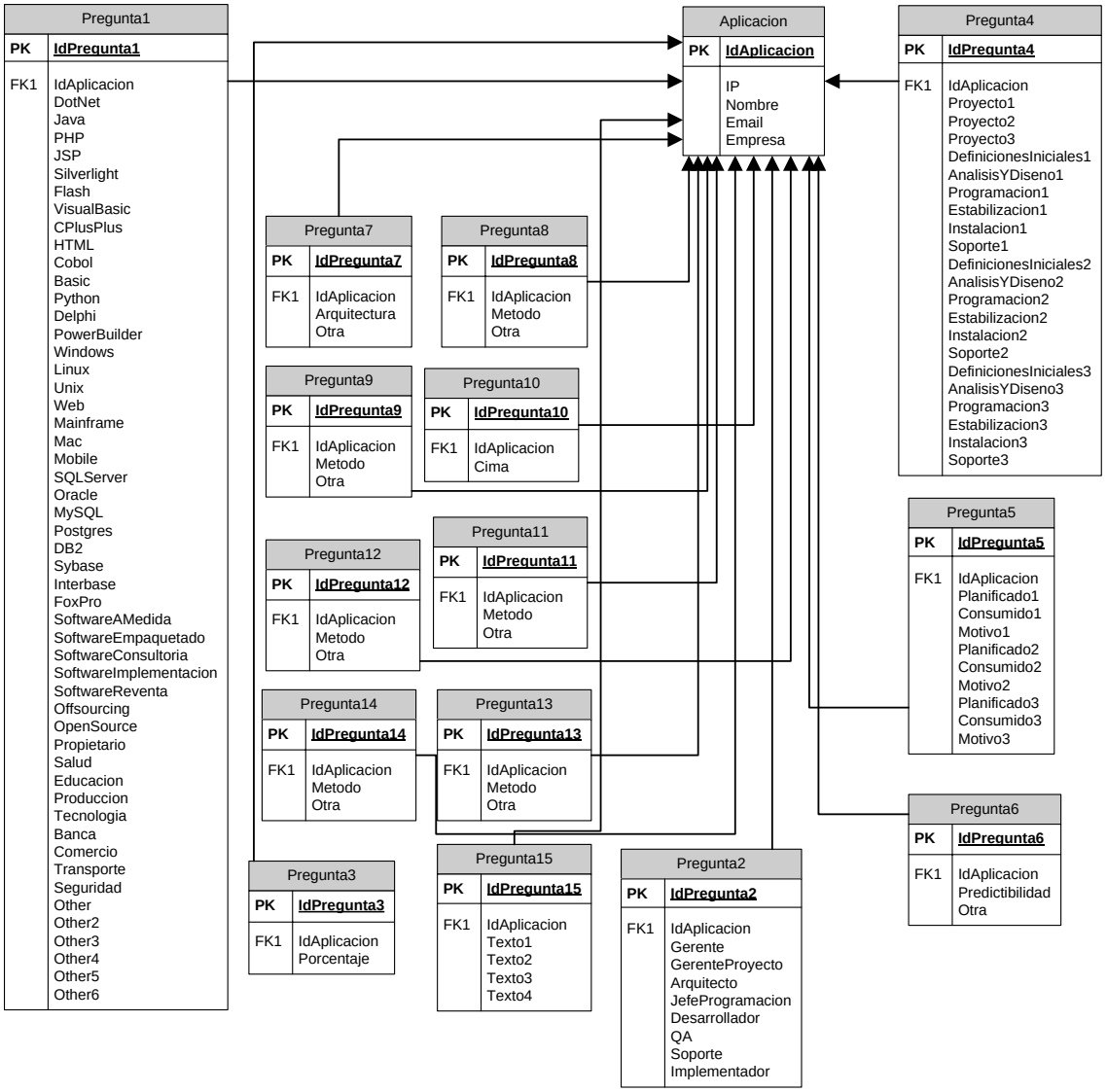
Para facilitar la difusión de las encuestas hacia los potenciales encuestados, en lugar de preparar un formulario en papel, se desarrolló un software preparado para funcionar en un entorno Web, de tal forma que cualquier persona pueda llenar las preguntas con una conexión a Internet, reduciendo la resistencia a participar en el proceso, y proporcionando facilidades como que pueda realizarse parcialmente o realizarse en cualquier momento ser llenada sin la presencia física de un encuestador.

La tecnología utilizada durante esta programación fue la siguiente:

- ASP.net 3.5
- Base de Datos SQL Server 2008
- Componentes de Telerik

Para almacenar la información de la encuesta, de tal manera que sea posible extraer posteriormente la información para realizar análisis, se incorporó una estructura de datos principal denominada “Aplicación”. Cada vez que una persona inicia la encuesta, registra los datos de la sesión en esta tabla, de tal manera que todas las siguientes preguntas quedan ligadas a la misma aplicación.

Este modelo facilitará principalmente la metodología de consultas cruzadas entre las diferentes preguntas. Se ha planteado el modelo de la base de datos de acuerdo al siguiente diagrama:



Como parte del presente estudio, se encuentra adjunto al presente documento, en medio electrónico, una carpeta denominada ANEXO 2, y dentro de ella una carpeta llamada “CodigoFuente”. En esta ubicación se encuentra todo el código fuente de la aplicación y los modelos de la base de datos involucrada.

El motor de encuestas fue publicado en los servidores web de SmartWork, en la siguiente dirección: http://encuesta.smartwork.com.ec

Ejecución de las encuestas

Selección del mercado objetivo

Uno de los objetivos de este estudio es el determinar las "mejores prácticas gerenciales" que se pueden aplicar en las empresas de software ecuatorianas, es por ello que para la ejecución de la encuesta se ha considerado como segmento objetivo a todas las personas que trabajan empresas ecuatorianas que se dedican al desarrollo de software o servicios afines.

Esto excluye a empresas que también brindan servicios tecnológicos, pero que no se encuentran directamente relacionadas con el desarrollo de software, tales como:

- Empresas de telecomunicaciones, conectividad, provisión de servicios tecnológicos y otras.
- Empresas especializadas en comercialización y venta de licencias de software, que su desarrollo haya sido hecho fuera del país.
- Empresas dedicadas a la provisión de equipamiento tecnológico, cableado estructurado, infraestructura de redes, servicios de redes y otros.
- Empresas de consultoría especializada, de implementación de software o cualquiera otra, que no cuenten con al menos un equipo interno de programadores.

En busca de un punto de partida para obtener el universo de investigación, se recopiló información publicada en la página Web del CONESUP (Consejo Nacional de Educación Superior)[47], y otras páginas web relacionadas[48] con la profesionalización en el área tecnológica, resultando en lo siguiente:

- En Ecuador existen 480 carreras aprobadas en el CONESUP, que están relacionadas con la Informática y Computación, tanto presenciales como a distancia, dentro de 75 universidades.
- No fue posible determinar el número exacto de títulos emitidos hasta la fecha en el Ecuador en la rama de Ingenierías en Informática, Computación, Sistemas y afines.
- En colegios profesionales tales como el Colegio de Ingenieros de Sistemas de Pichincha (CIISCP), cuentan solamente con 800 socios.

Luego de la primera investigación estadística, se determinó que no era adecuado determinar el universo utilizando los registros de los títulos profesionales universitarios emitidos en la rama, ya que aunque se hubiera contado con los datos exactos, no era posible determinar cuáles de las personas graduadas en el área de informática se dedican específicamente al desarrollo de software.

Es por ello que se consideró como mejor opción para determinar el universo muestral una investigación en registros de las empresas socias registradas en la AESOFT (Asociación Ecuatoriana del Software). Se encontraron más de 100

[47]Página Web del CONESUP: http://www.conesup.net/buscar_carreras_universidad.php?pagina=1 (revisada el 1 de Noviembre del 2010)
[48] Páginas Web: http://www.ciiscp.org (Colegio de Ingenieros de Sistemas de Pichincha) http://www.sidenorte.org (Sociedad de Ingenieros del Ecuador) (Todas consultadas el 1 de Noviembre del 2010)

empresas registradas, constatando que se encontraban presentas las más representativas de la industria ecuatoriana, y que a pesar que es evidente que no están registradas todas las empresas que realizan esta actividad[49], representa una buena base estadística para iniciar el estudio. Aún así probablemente la muestra no esté completa si no se considera que existen empresas de otros giros de negocio, que tienen equipos de programación dentro de su planta, y que probablemente presentan problemas similares a las empresas dedicadas al desarrollo de software.

En el Anexo 1 se presenta un grupo de empresas que se ha considerado importante que sean tomadas en cuenta para la investigación (de la lista original se ha retirado a discreción algunas empresas sin afectar el nivel de representatividad en el mercado Ecuatoriano al respecto del desarrollo de software [50]).

Luego de buscar información sobre cada una de las empresas señaladas en el universo de las 87 empresas presentadas, se encontró que varias de ellas funcionaban en más de una ciudad, pero principalmente se concentraban en Quito y Guayaquil, aunque muchas de ellas tienen operaciones en ciudades fuera del país. También se encontró que ciertas empresas que tienen

[49] De hecho, la empresa SmartWork no está registrada en esta asociación, pero se ha constatado la presencia de la mayoría de empresas cercanas y representativas del medio.

[50] Las empresas presentadas en la lista, fueron extraídas de la página web de la AESOFT (http://www.aesoft.com.ec/) en la sección de "Socios".
Se retiraron varias empresas que por mi experiencia personal en el mercado Ecuatoriano, se conoce que su giro de negocio principal no es el desarrollo de software, y por ende es conocido que no realizan proyectos de desarrollo de software de forma constante.

movimientos representativos en el sector, no se encontraban registradas en la base de datos de AESOFT. [51]

Para completar con la muestra, se realizó una investigación adicional que permitió determinar qué cantidad de equipos de programadores se han conformado en empresas en donde su línea principal de negocio "no sea" el desarrollo de software, tal como por ejemplo el sector bancario, de los bancos consultados[52] prácticamente todos tienen equipos de programadores para el desarrollo del software especializado. Esto también ocurre en empresas de diversos tipos de industrias, pero no se halló una estadística confiable que permita establecer un universo de investigación complementario.

Luego de analizar las diversas opciones encontradas para establecer el universo de investigación, he decidido trabajar de la siguiente manera:

- Universo de Investigación
    - 87 empresas registradas en la AESOFT[53]
- Muestra de la encuesta:
    - 15 empresas registradas en la AESOFT[54] (alrededor del 17% del universo de investigación)

[51] También se encontraron diversas empresas registradas en la Cámara de Comercio de Quito, que no se encontraban registradas en la AESOFT. http://www.lacamaradequito.com/index.php?option=com_wrapper&Itemid=6 (Cámara de Comercio de Quito)

[52] Por medio telefónico o por Internet pude contactarme con personas cercanas a las áreas de informática de los siguientes bancos ecuatorianos: Banco de Guayaquil, del Banco de Loja, del Produbanco y del Banco Solidario, quienes me confirmaron tener equipos de programación internos o bajo el modelo de outsourcing.

[53] Se refiere a las 87 empresas presentadas en el anexo señalado, por considerarse las más representativas en el mercado y las que pueden proveer y beneficiarse de mejor manera de las "buenas prácticas gerenciales". Hay que tener en cuenta que se eliminaron varias empresas por considerarse que no tienen trabajo constante en el desarrollo de software.

[54] Se utilizó una herramienta que permite calcular el tamaño de la muestra de forma estadística. Se utilizaron los siguientes parámetros: Intervalo de Confianza de 30, Población de 87, Nivel de Confianza

- 5 empresas que su giro de negocio sea diferente al desarrollo de software, pero que tengan este tipo de proyectos internamente de manera constante.[55]

- Planificación
  - Se buscará que por cada empresa sea posible que se llenen 3 encuestas, una enfocada al gerente de la empresa, otra al líder de proyectos y otra a los desarrolladores.
  - Por lo tanto se espera contar con alrededor de 60 encuestas llenas, como base mínima para el estudio estadístico.

Ejecución de Encuesta en Línea

Una vez que se hizo el levantamiento de la base de datos correspondientes al mercado objetivo seleccionado[56] y se elaboraron las plantillas de invitación por medio de correo electrónico, se abrió el ciclo de encuestas en dos etapas (sin alterar las preguntas planteadas para los dos grupos). La primera etapa fue orientada a los profesionales que se desenvuelven como programadores, y la segunda etapa fue orientada hacia la alta dirección y cargos de gerencia de proyectos.

La idea de dividir en ambas etapas daría mayor facilidad para el manejo del lenguaje y motivación para los diferentes grupos objetivo, y además permite hacer insistencia sobre las personas que no llenaron en la primera etapa.

de 99%. El resultado fue que el tamaño de la muestra debía ser de 15. La herramienta mencionada se encuentra en la siguiente dirección web: http://es.gmi-mr.com/resources/sample-size-calculator.php

[55] Durante la ejecución de las encuestas, se tratará de tomar contacto también con empresas alternativas que también cuentan con equipos de programadores, pero no se encuentran registradas en la AESOFT.

[56] Se lo hizo utilizando la información pública de las páginas web de las empresas seleccionadas, por referencias encontradas en el Internet y a través del contacto directo. El archivo de la base de datos utilizado se encuentra adjunta al presente documento con el nombre "BaseEncuestas.xlsx"

Una vez ejecutados ambos ciclos, se logró contar en la base de datos con 89 encuestas iniciadas, de las cuales solamente 39 fueron llenadas completamente[57]. La distribución con respecto a las preguntas contestadas, puede ser evidenciada en el siguiente gráfico:

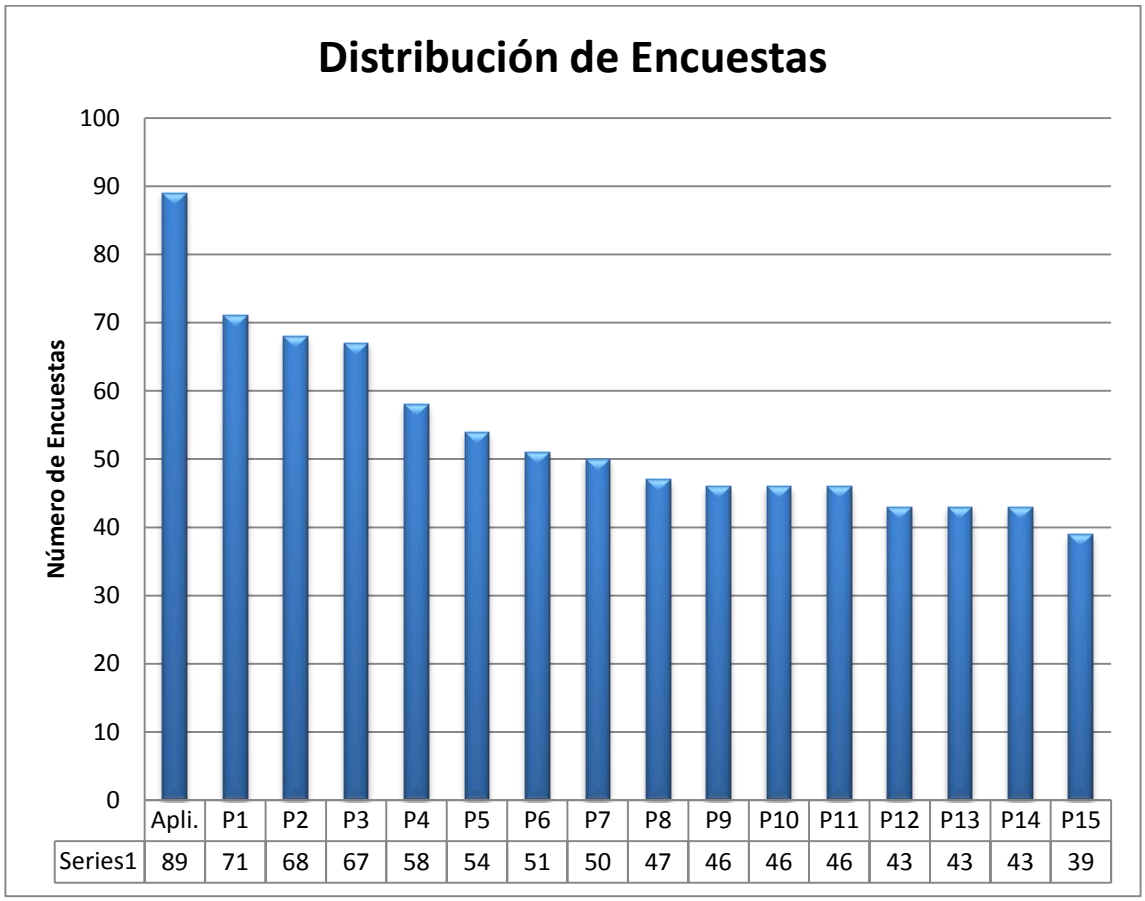


Figura 20: Encuesta: Distribución de las respuestas

[57] Hay que recordar que al ser una encuesta en línea, los participantes pueden retirarse del proceso en cualquier momento. Esto ocasionó que la mayoría haya contestado las primeras preguntas, pero no todos completaron la información de las últimas preguntas.

Análisis estadístico de los resultados

En base a las aplicaciones llenas registradas en el software de encuestas por Internet, se ha obtenido los siguientes resultados estadísticos: (Como complemento al análisis estadístico realizado en la encuesta planteada para el presente estudio, se tuvo la oportunidad de analizar un estudio publicado por la consultora Axentian, de origen argentino, llamado "Uso de la TI en Empresas de América Latina", publicado en Marzo del 2011, que contiene algunos análisis de tendencia similares a los planteados en esta investigación. Es por ello que como parte de este estudio se presentarán también un análisis comparativo del sector ecuatoriano, en comparación con la región)

Pregunta 1:

LENGUAJE DE PROGRAMACIÓN

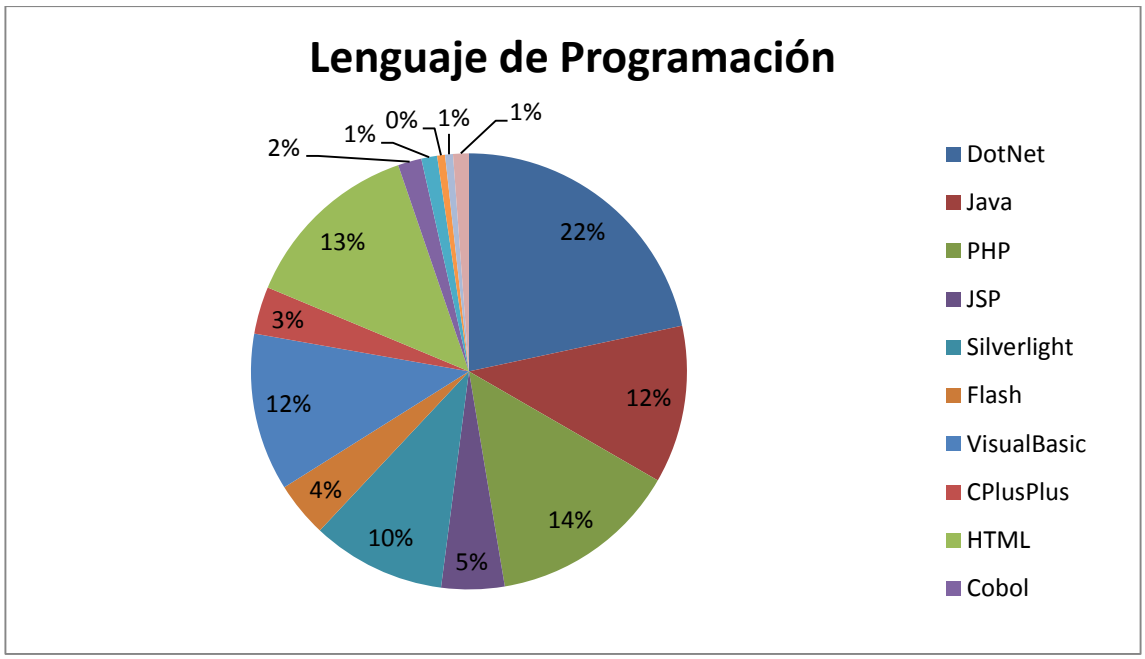


Figura 21: Encuesta: Tendencia de los Lenguajes e Programación

En base a este gráfico podemos darnos cuenta que existe una distribución relativamente equitativa entre los lenguajes más populares, con una tendencia mayor hacia el lenguaje .net.

Esto es positivo porque basado en esta información puede aseverarse, que los encuestados no solamente prefieren una sola tendencia teórica, sino que la muestra permitió que representantes de las diferentes filosofías fueran analizados[58].

También se confirma que los lenguajes de programación antiguos son ya poco usados en entornos profesionales.

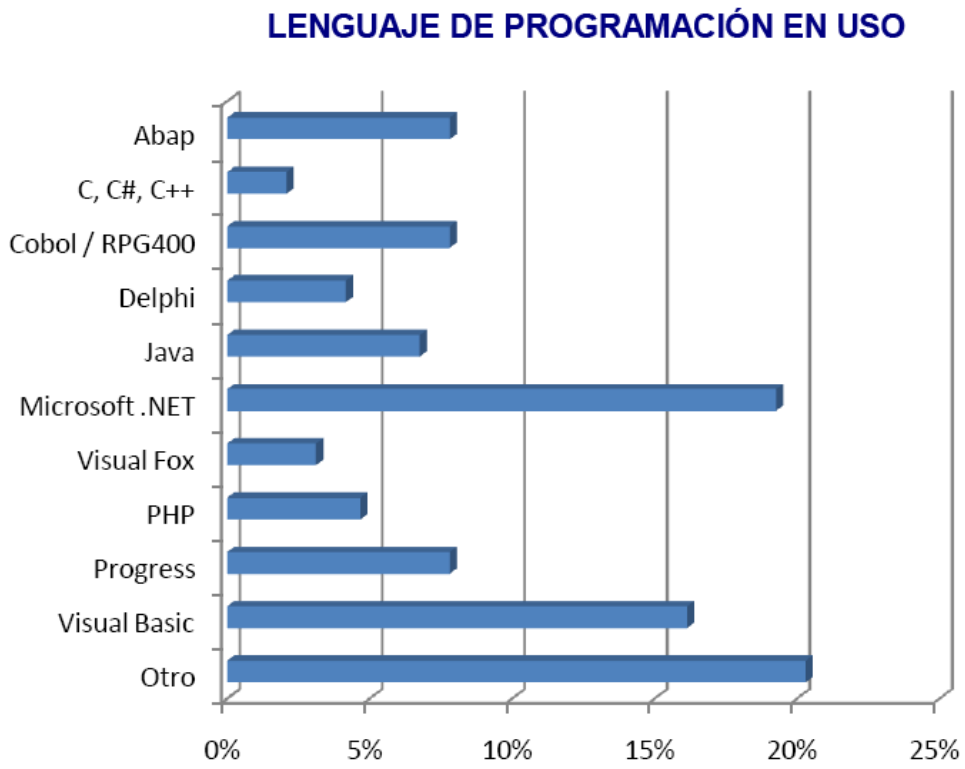


Figura 22: Distribución de Lenguajes de Programación para América Latina (Axentian, 2011)

[58] Una de las preocupaciones al momento de ejecutar la encuesta fue el hecho que SmartWork y conocidos cercanos trabajan con la tecnología .net, quienes fueron los primeros invitados a llenar la encuesta. Sin embargo, fue un trabajo arduo el encontrar actores de otras tendencias que también participen en el proceso de investigación, pues de lo contrario ocasionaría un sesgo en la información recopilada.

Basado en este gráfico podemos darnos cuenta que el Ecuador tiene varias tendencias similares a las presentadas en América Latina para los lenguajes más representativos, sin embargo en el país no se usa tanto las herramientas Cobol / RPG y Progress, que en la región aparecen tener bastante presencia.

PLATAFORMA

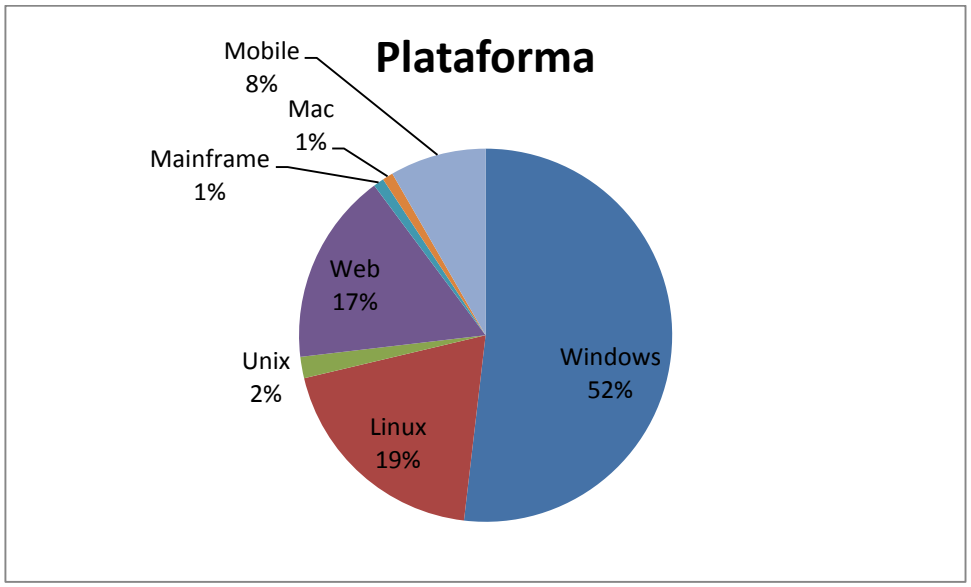


Figura 23: Encuesta: Distribución por Plataforma

Basado en este gráfico puede aseverarse que la tendencia de desarrollo de software se mantiene hacia la plataforma Microsoft Windows como predominante (52%), seguida de una alta presencia de desarrollo de software para ambientes libres (Linux 19 %), que se considera se encuentra en incremento[59], y también un crecimiento importante al respecto de sistemas orientados hacia la Web[60]. La presencia del 8% en desarrollo de software para

[59] En análisis informales e independientes a este estudio, he visto como la distribución de los profesionales dedicados al desarrollo de software, cada día más, dan apertura al uso de tecnologías de código abierto en el Ecuador. Probablemente esta tendencia se mantenga al alza gracias al soporte otorgado por el gobierno nacional para este tipo de herramientas, pero habrá un punto de equilibrio en donde se definirán las líneas de preferencia.

[60] Hablando de los últimos dos años, los clientes que han tomado contacto con nuestra empresa para realizar cotización de nuestros servicios a medida, la gran mayoría ha hecho mucho énfasis en la preparación de los sistemas para la Web y entornos distribuidos por medio de Internet.

dispositivos móviles es una tendencia reciente, y que se encuentra en crecimiento, especialmente desde la aparición de dispositivos como Ipad, Iphone, Ipod y Windows Phone.

BASE DE DATOS

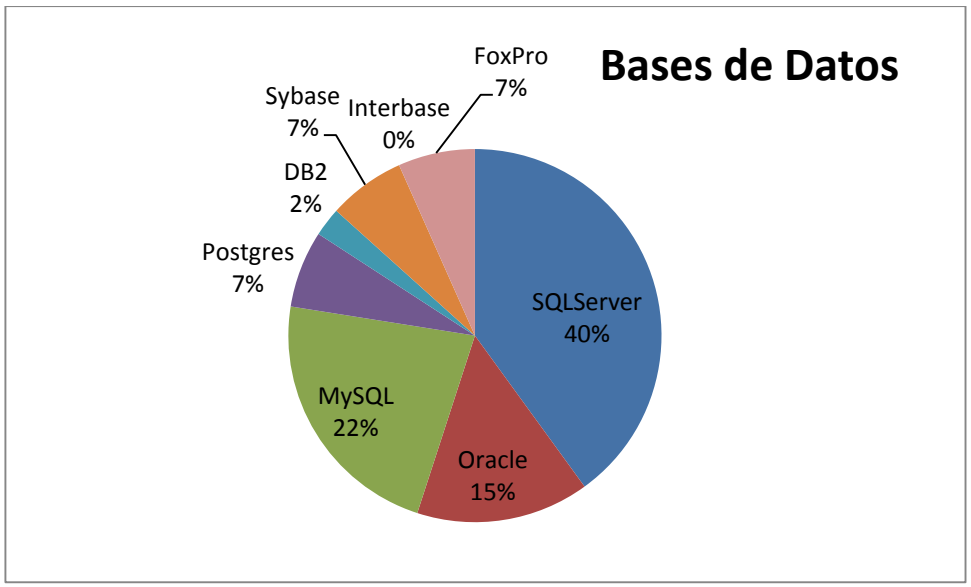


Figura 24: Encuesta: Tendencias de Uso de las Bases de Datos

Este gráfico nos permite aseverar que más de la mitad del mercado de bases de datos se mantiene con el uso de herramientas propietarias (Microsoft SQL Server y Oracle – 55%), sin desmerecer una presencia importante de las bases de datos de la tendencia del código libre, sumando un 29% (MySQL y Postgres).

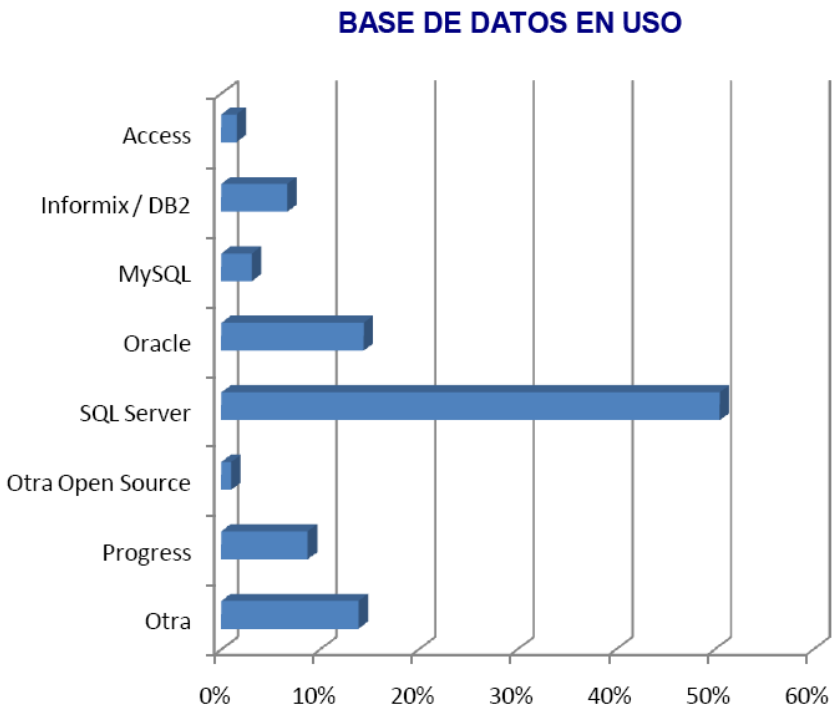


Figura 25: Grafico de tendencias de uso de Bases de Datos para América Latina (Axentian, 2011)

Al comparar las tendencias entre el estudio del medio ecuatoriano y el de América Latina, puede encontrarse bastantes similitudes, destacando que la herramienta Progress no es muy utilizada en el país.

MODELO DE COMERCIALIZACIÓN

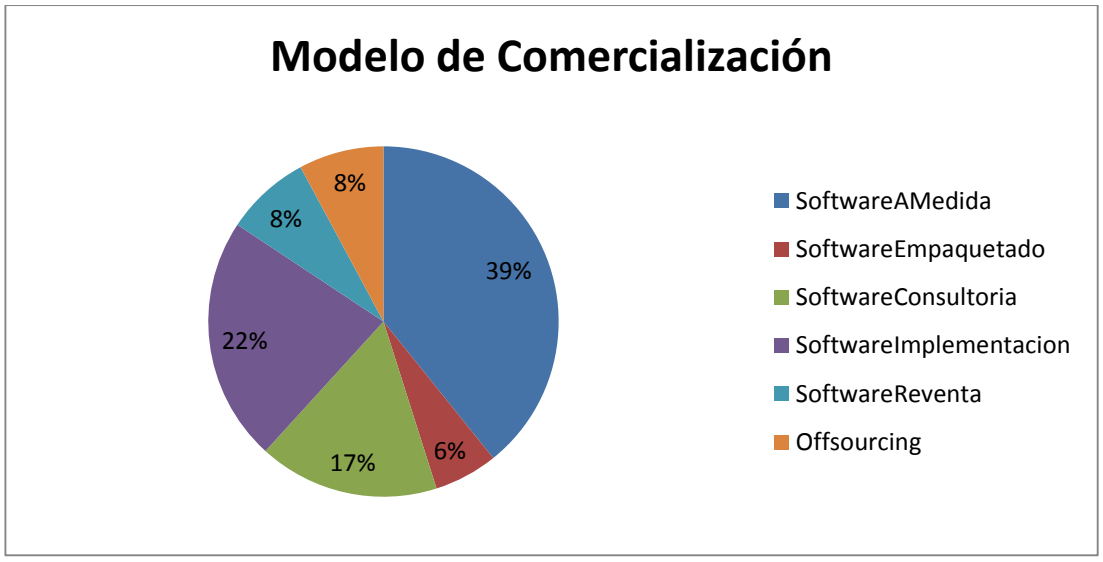


Figura 26: Encuesta: Modelo de Comercialización

Basado en este gráfico podemos encontrar que la mayoría de encuestados se encuentran en el negocio de Software a Medida (39%). Esto nos indica que una gran proporción de los negocios ecuatorianos dedicados al software utilizan este modelo en vista que es el que requiere de menores inversiones directas, ya que en la mayoría de estos proyectos se desarrollan en función del financiamiento del auspiciante de cada proyecto, y la empresa solamente dedica el capital intelectual.

Es novedoso que en este gráfico pueda encontrarse que el grupo de empresas dedicadas al software empaquetado sea muy bajo (6%), en comparación a los modelos a medida, consultoría e implementación. Esto denota que la mayoría del mercado de software está concentrado en servir las necesidades con trabajo profesional, y más no en el desarrollo de productos generalizados masivos.

ORIENTACIÓN

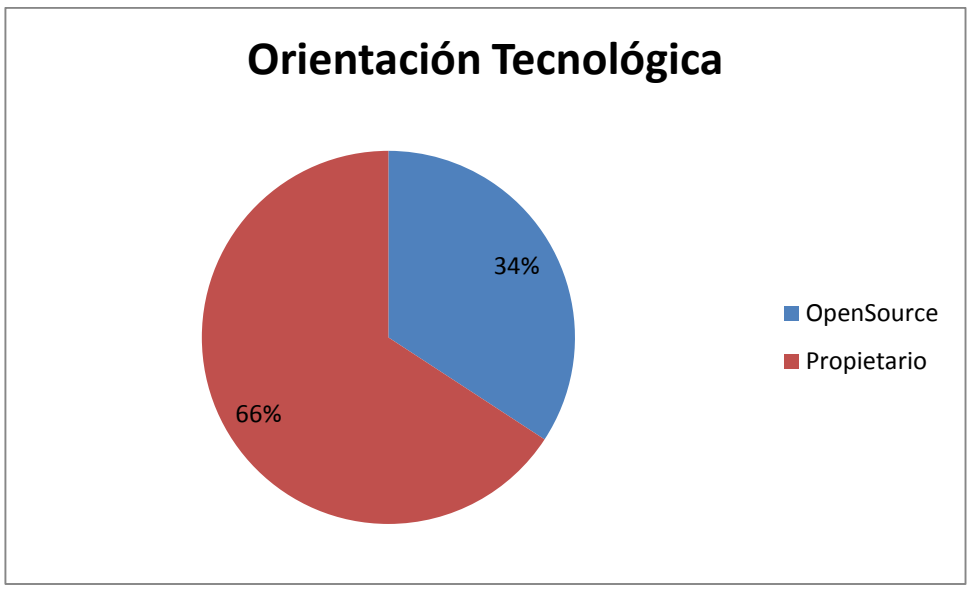


Figura 27: Encuesta: Tendencia de la Orientación Tecnológica

Este gráfico nos indica que la mayoría de profesionales del sector utilizan herramientas propietarias, sin embargo puede encontrarse que hay ya un gran mercado en la filosofía del código abierto. Se considera que este crecimiento en la tendencia de uso de herramientas libres corresponde fundamentalmente con la decisión del gobierno nacional de utilizar este tipo de herramientas como de uso oficial, abriendo casi todos los proyectos públicos con esta tecnología. Es por ello que el mercado se ha preparado para atender esta demanda, y se anticipa a decir que la proporción de uso de herramientas de código abierto seguirá subiendo paulatinamente en los siguientes años.

MERCADO VERTICAL

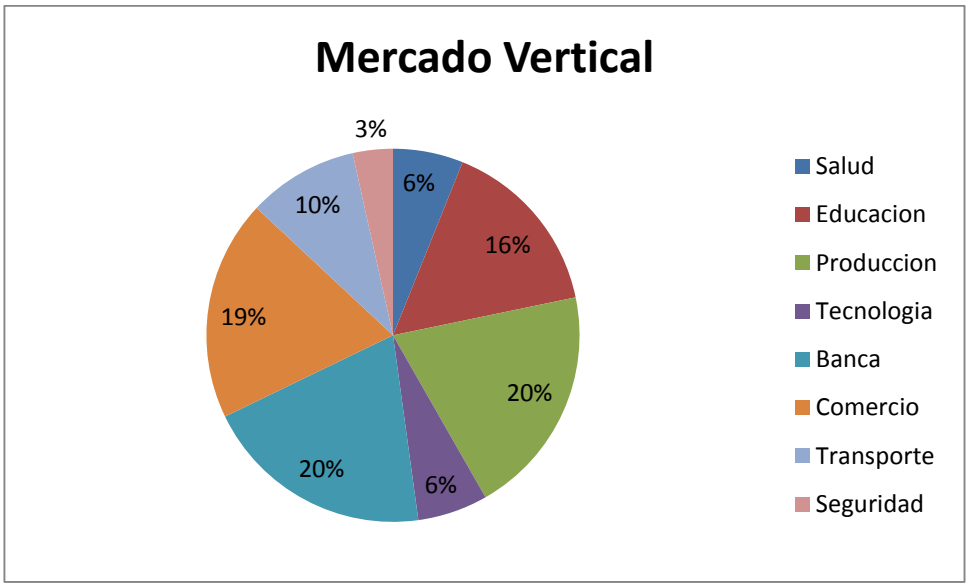


Figura 28: Encuesta: Tendencia del Mercado Vertical

Esta pregunta permite aseverar que la muestra seleccionada está equitativamente distribuida entre casi todas las opciones colocadas para el mercado vertical. Cabe destacar entonces que la mayoría de inversiones en desarrollo tecnológico se encuentran en la Banca (20%), las empresas comerciales (19%) y las empresas de producción (20%). Esta tendencia corresponde está conforme a lo estimado en la percepción personal.

Una de las preocupaciones más importantes al momento de ejecutar la encuesta, fue que se provoque un sesgo en la información por establecer la base de encuestados solamente entre los cercanos a las actividades del investigador.

Las respuestas obtenidas de la primera pregunta permite asegurar que los encuestados no están trabajando en un solo sector, con un solo lenguaje de programación o enfocados a una sola tecnología. Por ende, lo importante es que se ha asegurado que la encuesta entregará información sin sesgos específicos.

Pregunta 2:

La segunda pregunta está enfocada en comprender los roles que juegan los profesionales dentro de los proyectos de software. Para esto se ha determinado la consolidación de resultados en tres gráficos:

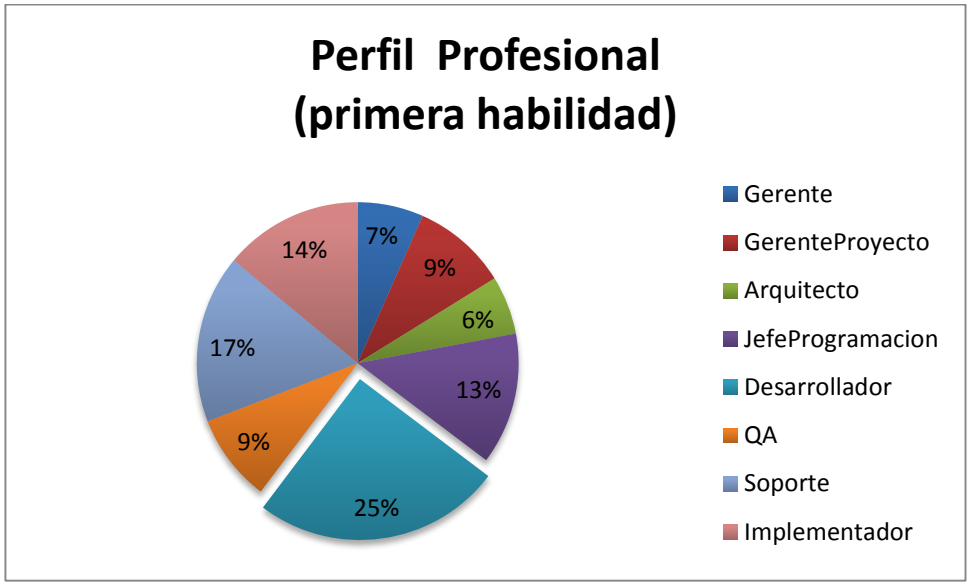


Figura 29: Encuesta: Tendencia del Perfil Profesional (primera habilidad)[61]

El primer gráfico nos permite señalar que una cuarta parte de los encuestados se dedican como actividad principal a las tareas de programación del software.

Así mismo puede verse que hay varias personas que cumplen los diversos roles dentro de un proyecto.

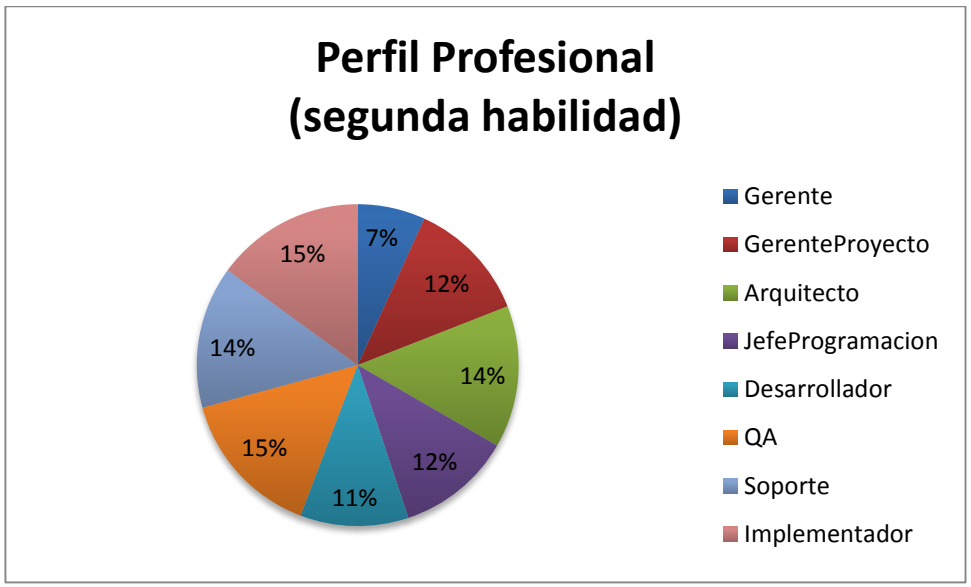


Figura 30: Encuesta: Perfil Profesional (segunda habilidad)[62]

[61] En el gráfico se ha utilizado la sigla QA (Quality Assurance), se refiere a las tareas de pruebas y aseguramiento de la calidad del software.

Como actividades de segunda prioridad para los encuestados, se puede encontrar que la muestra se distribuye con equitativamente para todas las opciones presentadas, disminuyendo considerablemente las actividades de desarrollo.

Esto da a entender que los desarrolladores tienen facilidades para asumir otro tipo de roles dentro del proyecto, pero los gerentes y arquitectos de software normalmente no asumen las tareas de programación de forma directa.

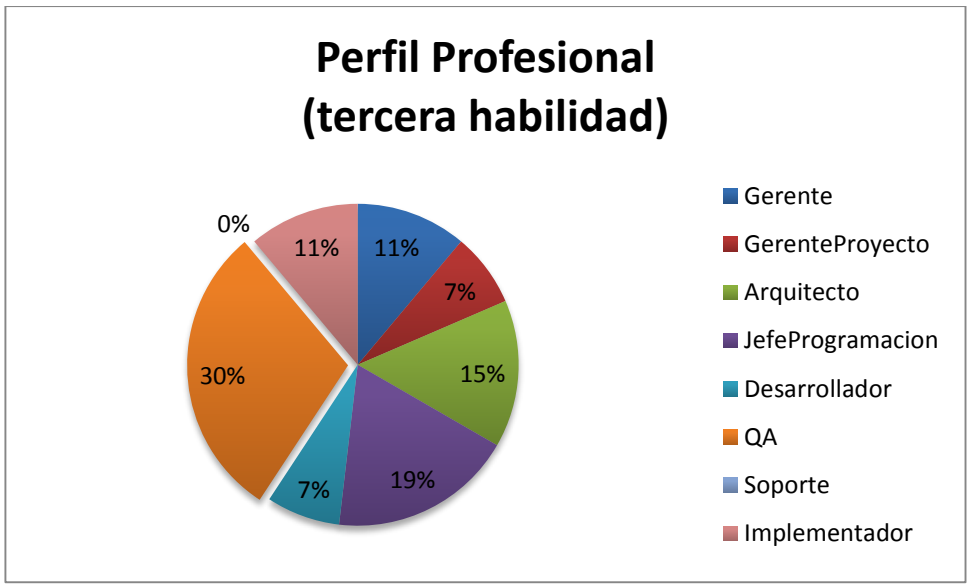


Figura 31: Encuesta: Perfil Profesional (tercera habilidad) [63]

Es importante determinar que la mayoría de los profesionales encuestados señalaron que su conocimiento y rol dentro de los proyectos de software correspondía a la tarea de QA (Quality Assurance). Esto es importante recalcar, que las empresas ecuatorianas tienen siempre la perspectiva de realizar pruebas a su software, pero no como actividad primordial.

[62] En el gráfico se ha utilizado la sigla QA (Quality Assurance), se refiere a las tareas de pruebas y aseguramiento de la calidad del software.
[63] En el gráfico se ha utilizado la sigla QA (Quality Assurance), se refiere a las tareas de pruebas y aseguramiento de la calidad del software.

Pregunta 3:

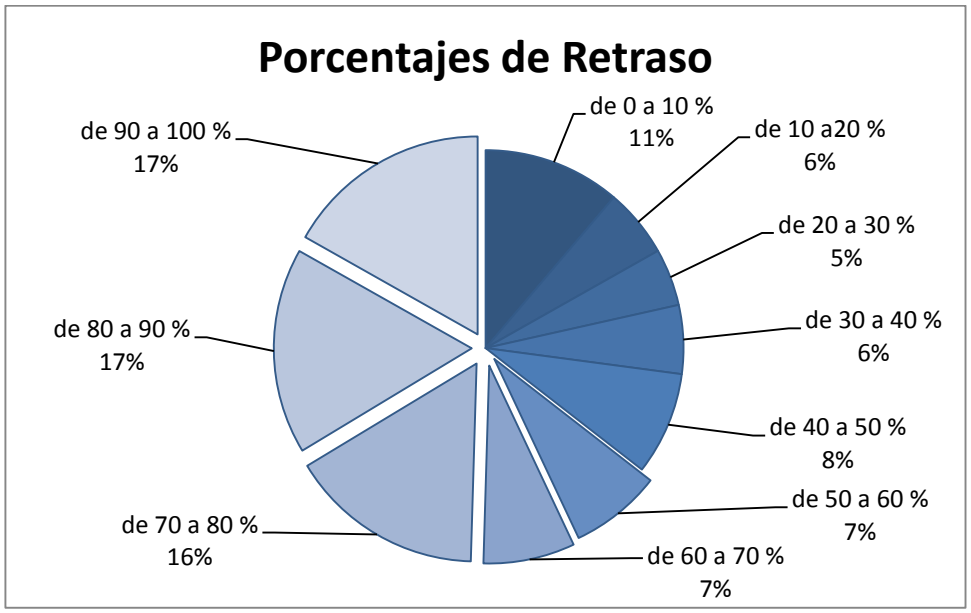


Figura 32: Encuesta: Tendencia de Porcentajes de Retraso

Una de las hipótesis planteadas para la presente investigación señala que las empresas de software tienen constantemente problemas con los retrasos.

Luego de analizar los resultados de esta pregunta, podemos sustentar y aseverar que la hipótesis planteada no solamente que es real, sino que es preocupante.

Este gráfico entrega las siguientes estadísticas:

- Más de la mitad de los encuestados opina que entre el 70% y 100% de los proyectos que ha participado se ha atrasado su cronograma, es decir que tienen retrasos muy constantemente.
- Más del 65% de los encuestados opina que al menos la mitad de los proyectos de software que han participado se atrasan.

- Menos de un cuarto de los encuestados considera que menos del 30% de sus proyectos se atrasan.
- Se encuentra que solamente el 11% de los encuestados tienen una metodología que les permite tener pocos retrasos.

Esto significa que los atrasos en los proyectos de software para la mayoría de empresas es crónico y repetitivo. Es por ello que a continuación se decidió elaborar una curva de probabilidades del retraso, considerando cada uno de los valores individuales de la muestra, y determinando una curva de Distribución Normal[64] que nos permita establecer las probabilidades de retraso en el mercado ecuatoriano.

La muestra establecida nos indica que la media del retraso se encuentra en el 65%, con una desviación estándar de 30%, presentando una curva de Distribución Normal de la siguiente manera:

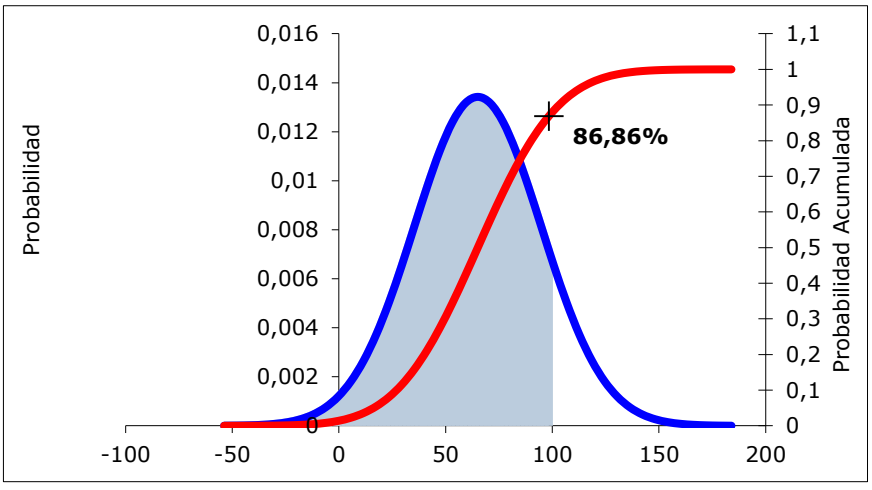


Figura 33: Encuesta: Curva de Gauss al respecto de los retrasos de los proyectos de software

[64] Para la elaboración de las curvas de distribución normal, se determinó una media de 65.08 y una desviación estándar de 29.72. Si se desea revisar con mayor detalle, en el archivo de Excel adjunto llamado "NormalDistribution.xls", se puede encontrar los cálculos utilizados para este análisis.

Luego se hizo un análisis de probabilidades bajo la curva de Gauss, de acuerdo al siguiente gráfico:

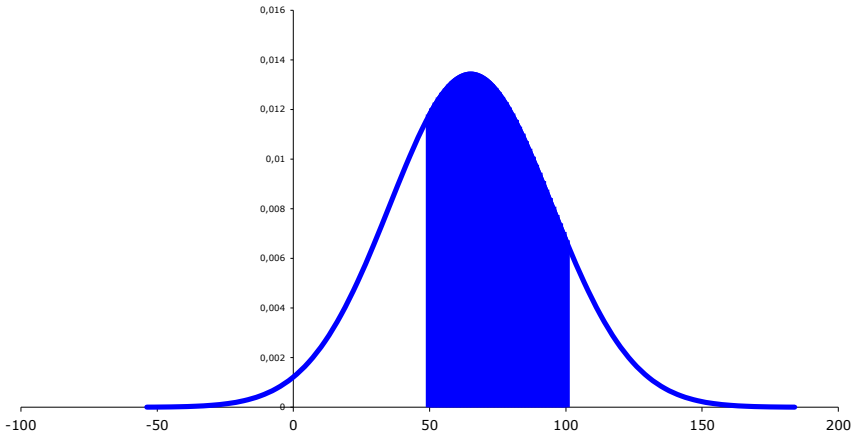


Figura 34: Encuesta: Probabilidad de que un proyecto se atrase

$Pr(x_{min}<x<x_{max})$ 57.41%

Este análisis nos permite determinar una cifra preocupante, que indica que la probabilidad de que el software tenga un retraso se ubique en el 57.41%.

Esta información nos ha permitido no solamente validar que los retrasos en este tipo de proyectos son reales y cuantificables, sino que como industria ecuatoriana, los valores de incidencia son altos y corresponden a un problema crónico.

Pregunta 4:

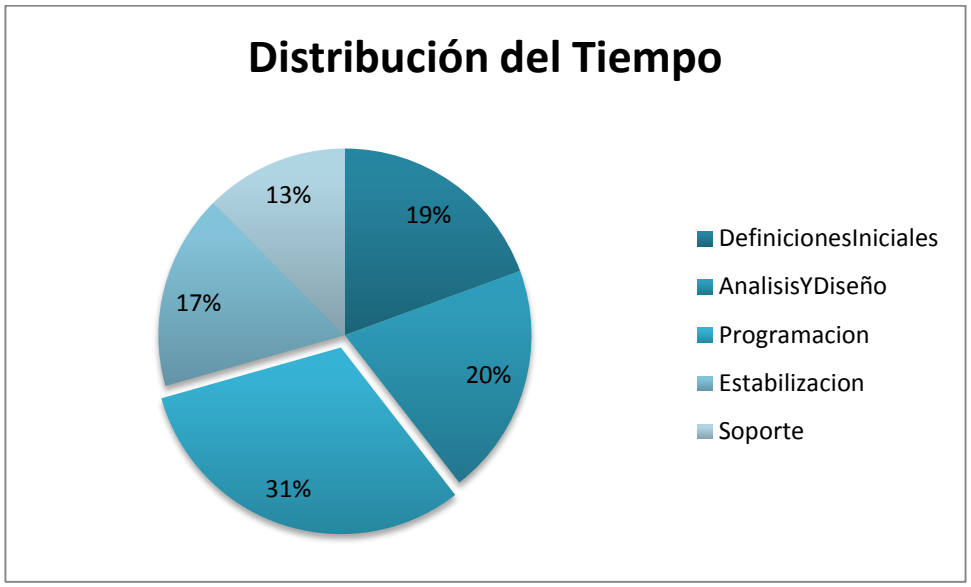


Figura 35: Encuesta: Tendencia de la Distribución del Tiempo en las diferentes etapas

La orientación de esta pregunta fue el determinar si los equipos de desarrollo invierten mayormente en tareas de elaboración del producto final (programación), o en su diseño y conceptualización.

La encuesta nos permite aseverar que un porcentaje importante de los encuestados ha consumido mayor tiempo en el desarrollo de software, sin embargo, es muy notorio que en comparación con el tiempo total desplegado para las otras etapas, no es ampliamente mayor.

Adicionalmente en este gráfico, llama la atención el valor obtenido en la primera opción, denominada “definiciones iniciales” con un 19%. Esto quiere decir que hay mucha inversión de tiempo en la etapa donde se realiza la “traducción” de los requerimientos en palabras de los usuarios, a requerimientos técnicos, y a la larga en la conceptualización inicial de la solución, previo a su estudio formal. Se pensaría normalmente que una inversión de tiempo mayoritaria se encontraría en la segunda variable, que es

la referente al análisis y diseño, sin embargo, su diferencia con la primera prácticamente no es significativa.

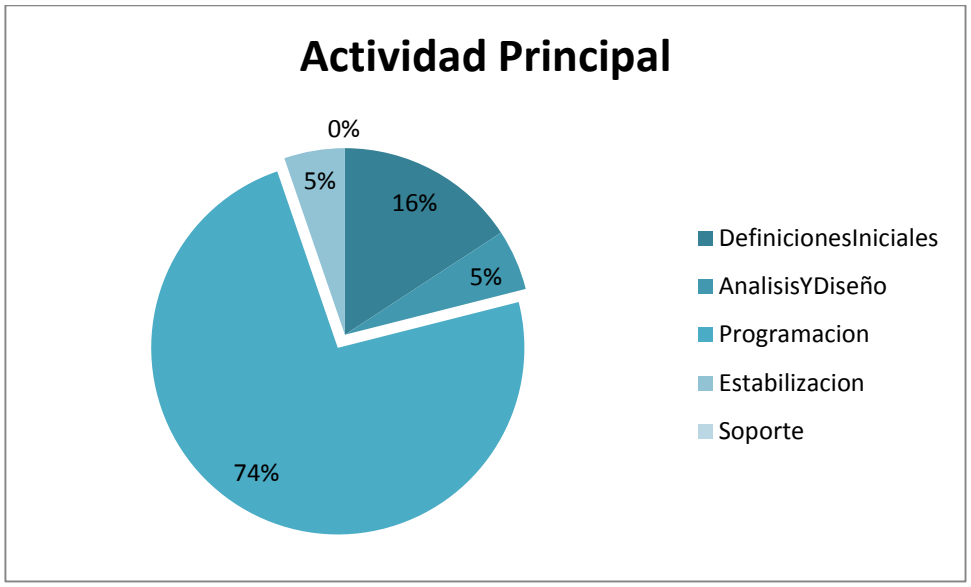


Figura 36: Encuesta: Etapa con mayor asignación de tiempo

Basados en los resultados del primer gráfico, se elaboró un nuevo gráfico analítico que nos permite visualizar a qué cantidad de encuestados les pareció la actividad de programación la más importante y que más tiempo les tomó.

El resultado de este gráfico, arroja un resultado interesante y no muy esperado[65], que es el hecho de que existe alrededor de un 25% de proyectos de software, en donde el tiempo utilizado para la programación del sistema es menor a otro tipo de actividades. Esto nos trae como conclusión que hay cierta variación entre las diversas empresas al respecto de esta asignación de tiempos y esfuerzos para cada una de las etapas.

[65] A pesar que las empresas de desarrollo de software invierten una alta cantidad de tiempo en análisis y diseño, en mi experiencia había visto pocos proyectos que hayan invertido más tiempo en estas tareas, que en la elaboración del producto en sí.

Pregunta 5:

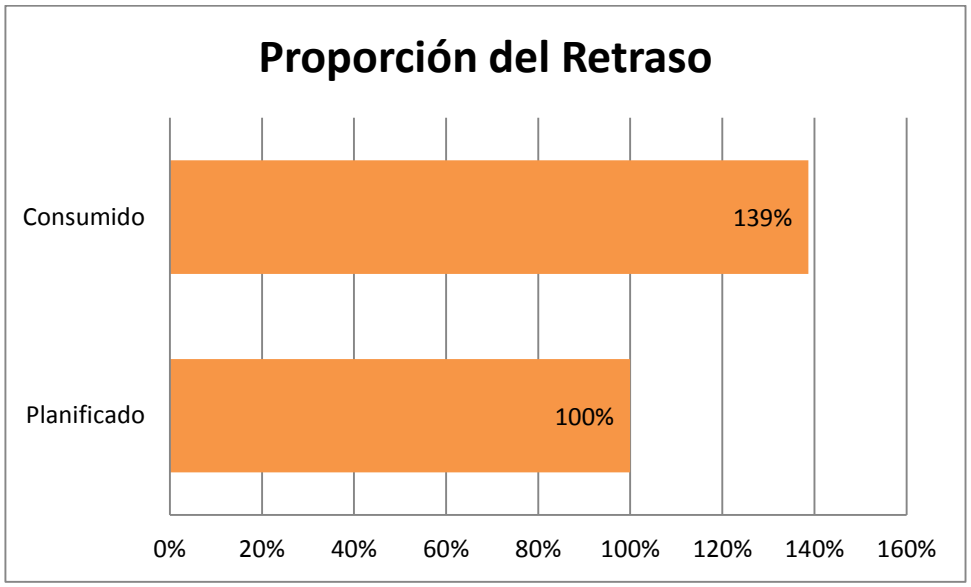


Figura 37: Encuesta: Magnitud del Retraso

En la pregunta 3, se pudo observar que la existencia de los atrasos en los proyectos de software en el Ecuador es un problema real y tangible, en tanto que la quinta pregunta fue orientada en determinar la profundidad y magnitud del retraso en los proyectos de software, llegando a demostrar lo siguiente:

El gráfico nos indica que en promedio, los proyectos que se atrasaron consumieron un 39% más del tiempo con respecto a lo planificado. Esta cifra es preocupante, porque indica no solamente que se trata de un problema real y tangible, sino que existe un margen de error muy alto, y que pone en clara duda las técnicas actualmente utilizadas en el mercado ecuatoriano para la medición de tiempos. Esto llevaría a pensar entonces que el nivel de confiabilidad del cumplimiento “a tiempo” de un proyecto es bajo y la confianza

de los clientes se está deteriorando constantemente por los amplios márgenes de error[66].

Esta afirmación también permitiría señalar que los márgenes de utilidad de las empresas del sector están siendo consumidos por los retrasos, que posiblemente sean el motivo de una reacción en cadena, formando un círculo vicioso: "el gerente del proyecto preocupado por el cumplimiento de los tiempos presiona a los desarrolladores y ofrece a los auspiciantes fechas considerando escenarios optimistas; los desarrolladores invierten más esfuerzo y desgaste en un punto del proyecto disminuyendo su nivel de productividad general; el proyecto avanza de forma descontrolada y se atrasa en el cronograma global; y el gerente de proyecto nuevamente presiona a los desarrolladores y negocia tiempos con los auspiciantes quienes ya no confían; los desarrolladores se esfuerzan más pero encuentran desmotivación, el proyecto se sigue atrasando más…"

[66] Este tipo de aseveraciones las he encontrado con relativa frecuencia entre los clientes más grandes, los cuales tienen un camino recorrido al respecto de fracasos en este tipo de proyectos. Tuve la oportunidad de tratar con un cliente, que puso una política interna de adquisiciones que señalaba que cualquier proyecto de software deberá ser desarrollado por cualquier empresa que no sea ecuatoriana, justamente por este tipo de problemas. Aún así, tampoco he podido constatar que las empresas extranjeras consigan ejecutar los proyectos en menor tiempo que las empresas ecuatorianas, es más, por el contrario les toma un poco más de tiempo.

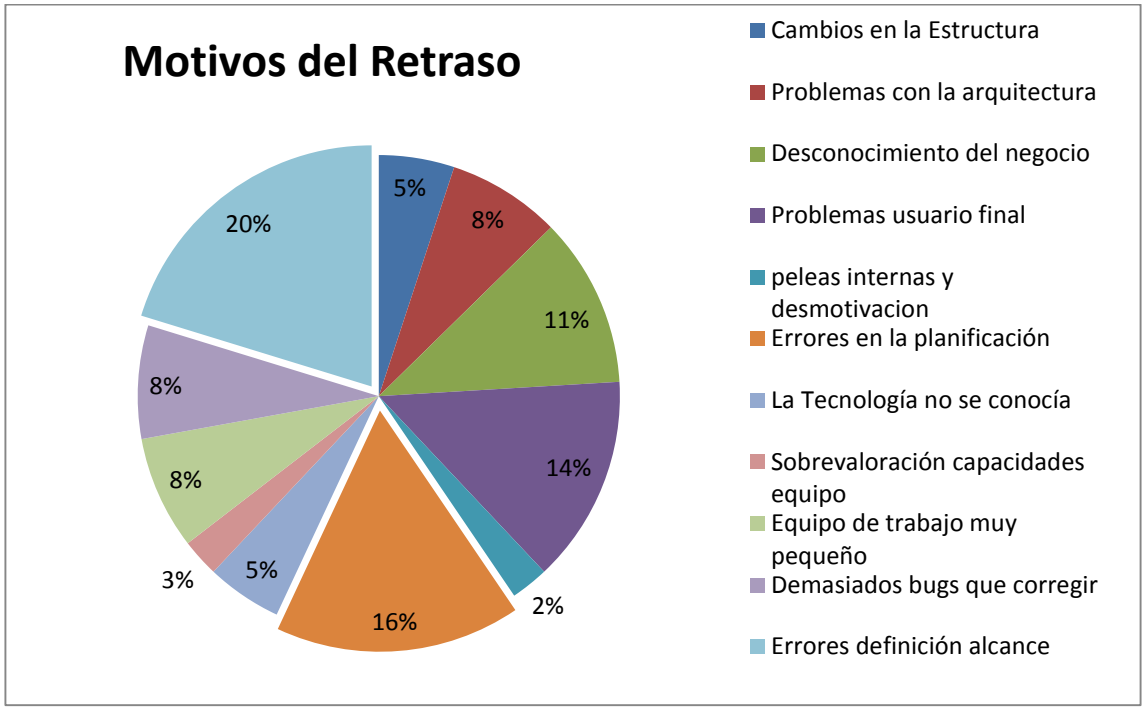


Figura 38: Encuesta: Motivos del Retraso

La segunda parte de la pregunta busca encontrar los motivos más recurrentes que ocasionaron los retrasos en los diferentes proyectos. El resultado es interesante, ya que se puede aseverar que la mayoría de los problemas se encuentran entre "Errores de definición del alcance" (20%), "errores en la planificación (16%)" y "desconocimiento del negocio (11%)", todas pertenecen al área del análisis y planificación (Total= 47%). La distribución porcentual de las demás opciones es dividida entre todas las opciones, permitiéndonos entender que los problemas planteados son reales y se presentaron en varios proyectos. Probablemente si se mitigaran los motivos más frecuentes, se obtendría buenas prácticas comunes, que pueden ser aplicadas en diversos tipos de empresa.

Pregunta 6:

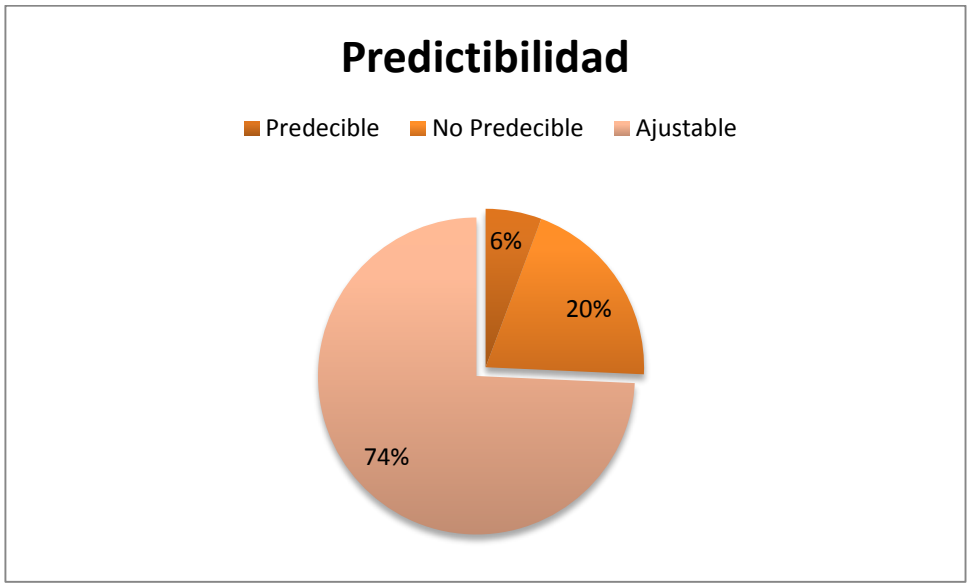


Figura 39: Encuesta: Predictibilidad del tiempo del Software

Al analizar el gráfico resultante de la pregunta 6, se puede aseverar que la mayoría de los encuestados considera que el software no puede ser medido exactamente por un algoritmo matemático, y que confía en sus habilidades de intuición y de la experiencia para determinar las métricas a utilizarse en la planificación del proyecto.

Es interesante observar que el 20% de los encuestados cree que el software "no es predecible", es decir, que se entendería que hay un alto grupo de personas que consideran que el software no puede planificarse bajo los métodos "tradicionales" utilizados para ejecución de proyectos de otras áreas. Con esto también se valida la hipótesis planteada al inicio del estudio que señalaba que el software es un proceso de mejora continua, en lugar que un producto; ya que solamente el 6% de los encuestados cree que puede medir exactamente el alcance del proyecto de forma certera.

Pregunta 7:

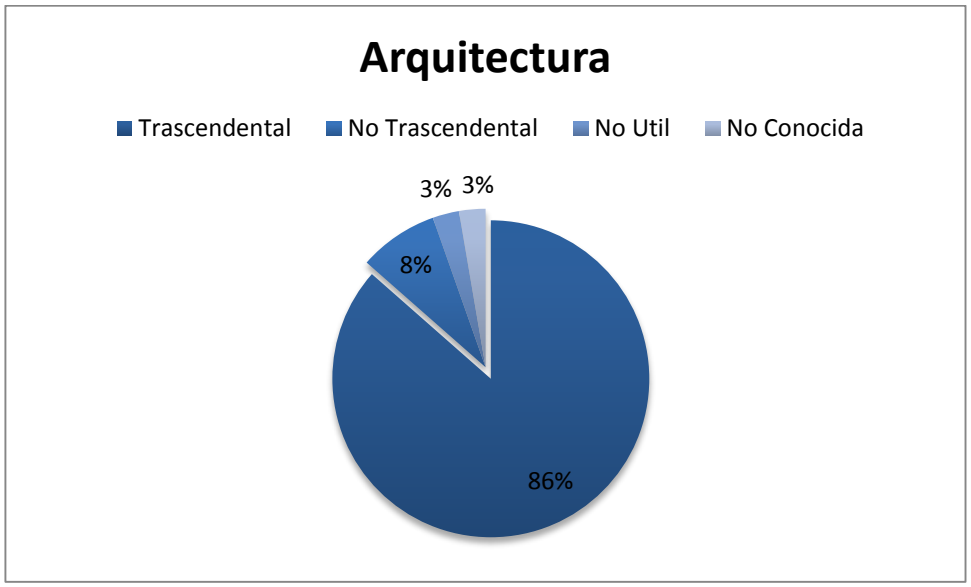


Figura 40: Encuesta: Uso de Arquitectura de Software

En este gráfico es posible visualizar una contundente tendencia de los encuestados al respecto de que la arquitectura de software es “trascendental”. Esto quiere decir que a pesar que existen retrasos en la planificación del software, la industria ecuatoriana está preocupada por la utilización de estándares globales y la elaboración de estudios formales sobre la tecnología a aplicarse.

Esta estadística es positiva para la industria ecuatoriana, porque demuestra que las empresas están preocupadas por el uso y creación de nuevas tecnologías, lo que presenta un escenario técnico muy atractivo a vista del mercado internacional del software. Es posible que ese sea el justificante del crecimiento de las empresas multinacionales presentes en el país en los últimos 3 años.

Pregunta 8:

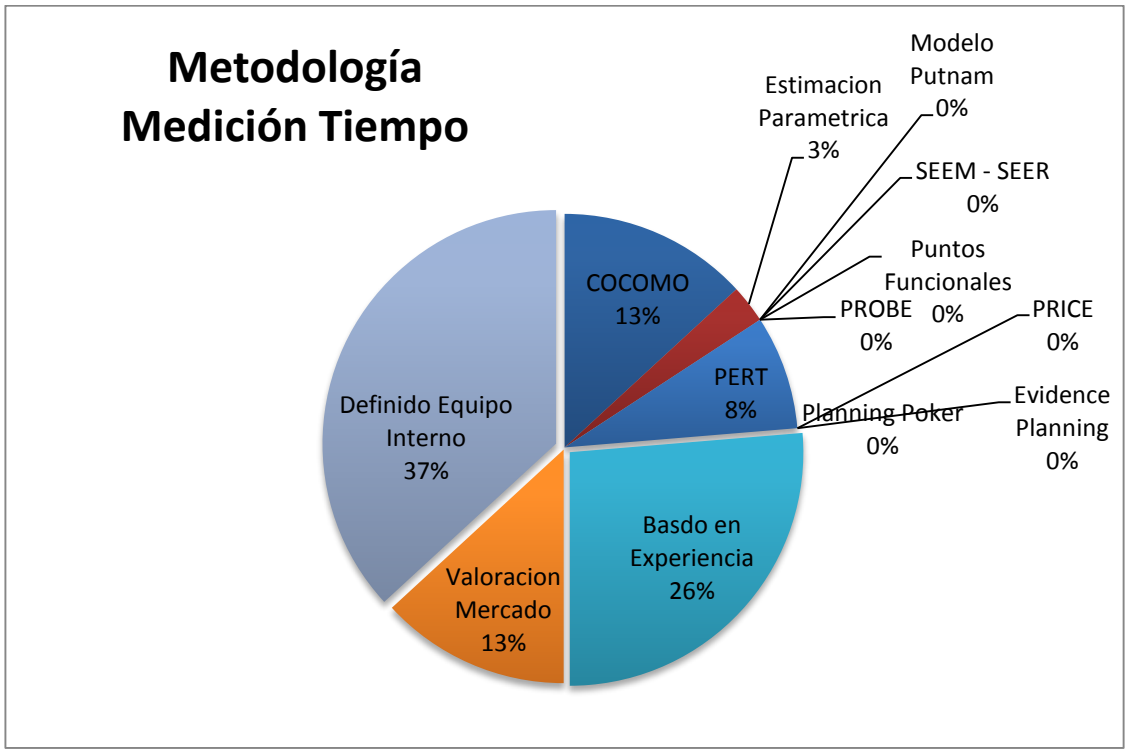


Figura 41: Encuesta: Metodología de Medición de Tiempo

Esta pregunta está orientada a complementar la información de la pregunta 6, en la que se demostró que la mayoría de personas utiliza métodos de ajuste basados en la intuición y experiencia, para realizar la medición de tiempos.

En esta pregunta se corrobora que el 76% de los encuestados utiliza métodos basados en la intuición y experiencia, en lugar que métodos que cuenten con un sustento teórico. Llama la atención que en este grupo, la mayoría se ha orientado a un análisis con el equipo de trabajo. Se considera que el definir los tiempos directamente con los desarrolladores cae en el escenario de exceso de optimismo, lo que trae la elaboración de cronogramas muy optimistas y difíciles de cumplir.

Además, en el gráfico presentado encontramos que los métodos matemáticos más utilizados son COCOMO (13%), PERT (8%) y Estimación Paramétrica (3%). El resto de métodos no fueron utilizados por ningún encuestado, lo más probable es que no se los conocía previamente[67].

Pregunta 9:

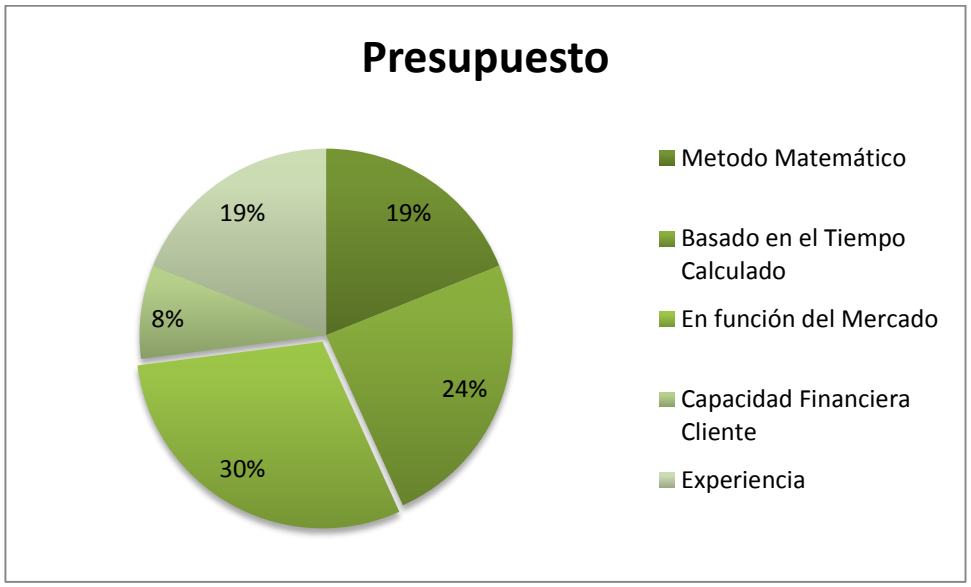


Figura 42: Encuesta: Método de valoración del Presupuesto

Los resultados de la pregunta 8 se corroboran en la pregunta 9, señalando que las personas que utilizaron la experiencia o intuición como base para medición del tiempo, también lo hicieron para el establecimiento de presupuestos.

Es importante señalar que el mayor grupo de encuestados utiliza los presupuestos en función del mercado, es decir que su fundamento parte de la percepción de valor en comparación con la competencia, y son un porcentaje menor los que utilizan mecanismos basados en sustento matemático.

[67] Otro de los motivos del desconocimiento de varios algoritmos matemáticos puede deberse a que varias son herramientas pagadas y con métodos de licenciamiento de valores restrictivos, que imposibilitarían su acceso.

Pregunta 10:

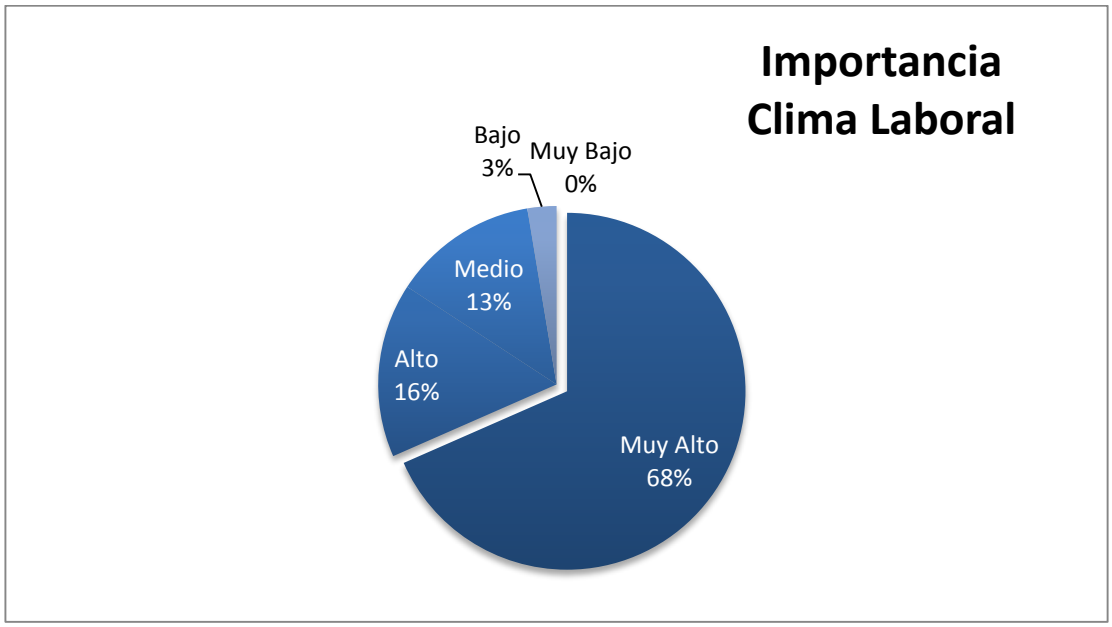


Figura 43: Encuesta: Importancia del Clima Laboral

Es amplio el margen de encuestados que afirman que contar con un clima laboral adecuado es indispensable para lograr los objetivos del desarrollo de software.

Esta afirmación llevaría a reflexionar si es que las empresas ecuatorianas de software se han preocupado por mejorar su clima laboral interno y sus consecuencias en la productividad de sus empleados, que son a la larga, factores que inciden en algunos casos de forma directa en la generación de retrasos[68].

[68] Tuve la oportunidad de participar parcialmente en un proyecto de software en donde el arquitecto asignado defendía hasta con agresividad su planteamiento tecnológico sin aceptar sugerencias de revisión y mejora, formando los grupos de defensores y detractores. La consecuencia fue un clima laboral tan negativo que muchos se retiraron, se difuminó la concentración y el proyecto en general se retrasó ampliamente.

Pregunta 11:

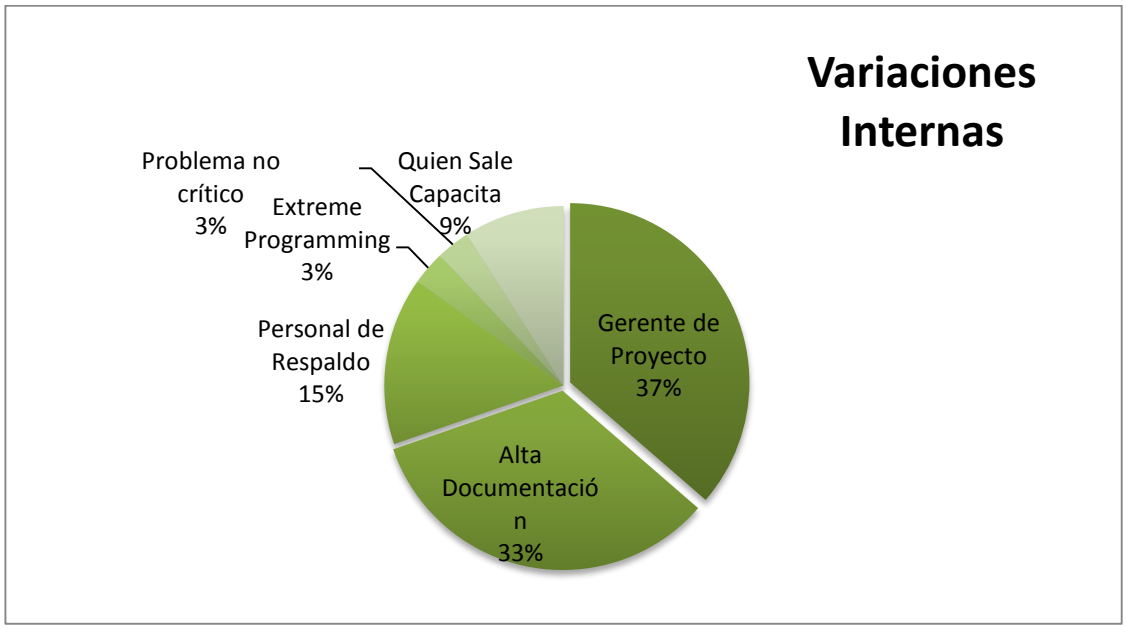


Figura 44: Encuesta: Impacto de las Variaciones Internas del Equipo de Trabajo

Las tendencias más marcadas son las de asignación de un gerente de proyecto profundamente involucrado en el código fuente, la de creación de una documentación muy detallada y contar con personal de respaldo, opciones las cuales representan un rubro importante en el costo del proyecto al ser tareas que deben ser llevadas por profesionales “no programadores”, llevándonos a la conclusión de que las empresas están actualmente invirtiendo importantes recursos en la gestión del conocimiento adquirido en los proyectos.

Las respuestas a esta pregunta también orienta a afirmar que el cambio de una persona en el equipo de trabajo impacta en el retraso del proyecto y las empresas conscientes de este problema realizan diversas estrategias de mitigación.

Pregunta 12:

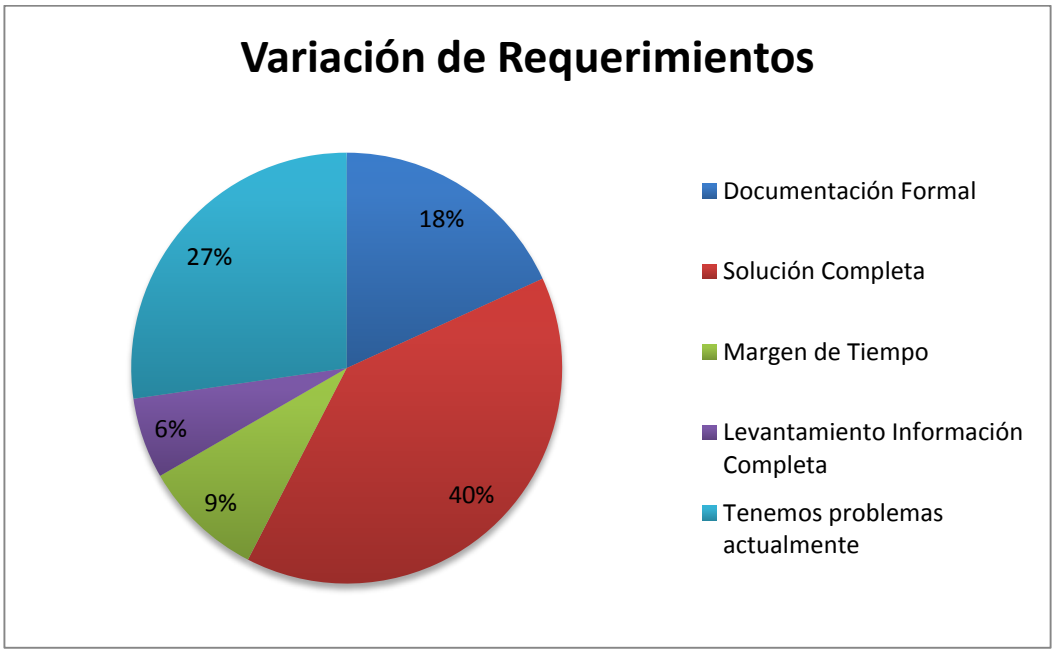


Figura 45: Encuesta: Tendencia de la Variación de Requerimientos

Las respuestas a esta pregunta denotan una realidad interesante, la mayoría de empresas no son estrictamente inflexibles con respecto a la documentación formal, y se reconoce que ser muy flexibles ocasionaría también el fracaso del proyecto, por lo tanto se podría afirmar que los proyectos de software en el Ecuador deben ser preparados para flexibilizar sus alcances en función del tiempo, pero basados en negociación de tiempos. Esto nos permite deducir que en lugar de establecer una metodología de medición exacta, es mejor plantear buenas prácticas al respecto las tácticas de negociación utilizadas sobre la variación de requerimientos.

Las respuestas a esta pregunta también presentan una realidad preocupante, el 27% de los encuestados considera que se “tienen problemas actualmente”, significando la insuficiencia de estrategias válidas para mitigarlo, dejando en evidencia que el mercado nacional tiene deficiencias teóricas y prácticas en

este tema. Esto da un punto de entrada importante al respecto de la profundización de buenas prácticas.

Pregunta 13:

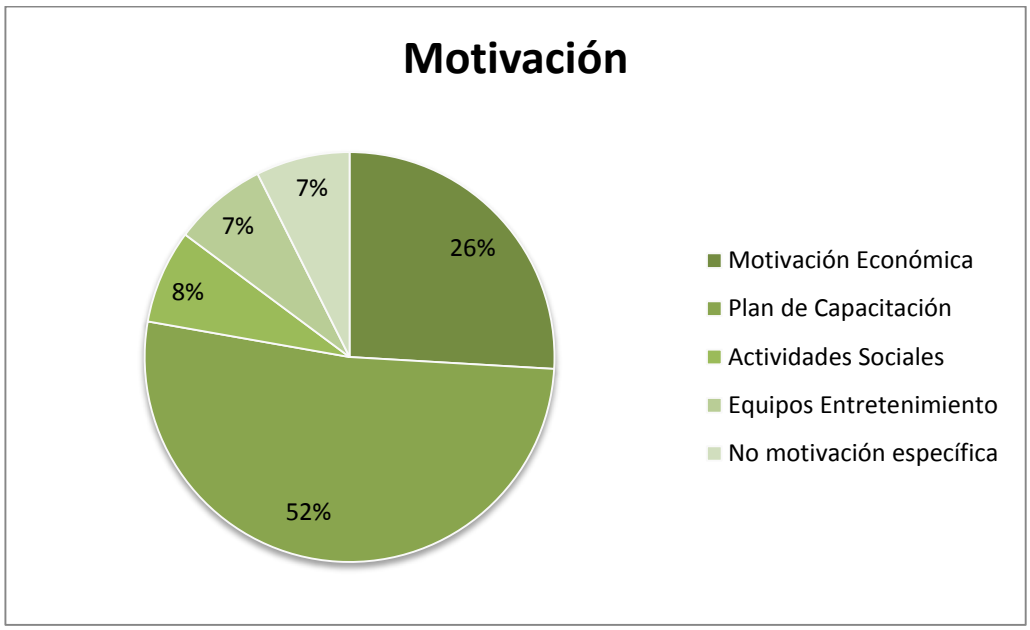


Figura 46: Encuesta: Tendencia sobre los métodos de motivación al personal

Esta pregunta está orientada a complementar la información entregada por la pregunta 10, sobre de la importancia del clima laboral. En este gráfico se encuentra un resultado muy interesante: la motivación económica es importante, pero mucho más relevante es el establecimiento de un "Plan de Capacitación" que permita a los equipos técnicos progresar en sus habilidades profesionales y técnicas, y estar siempre a la vanguardia de la tecnología.

Pregunta 14:

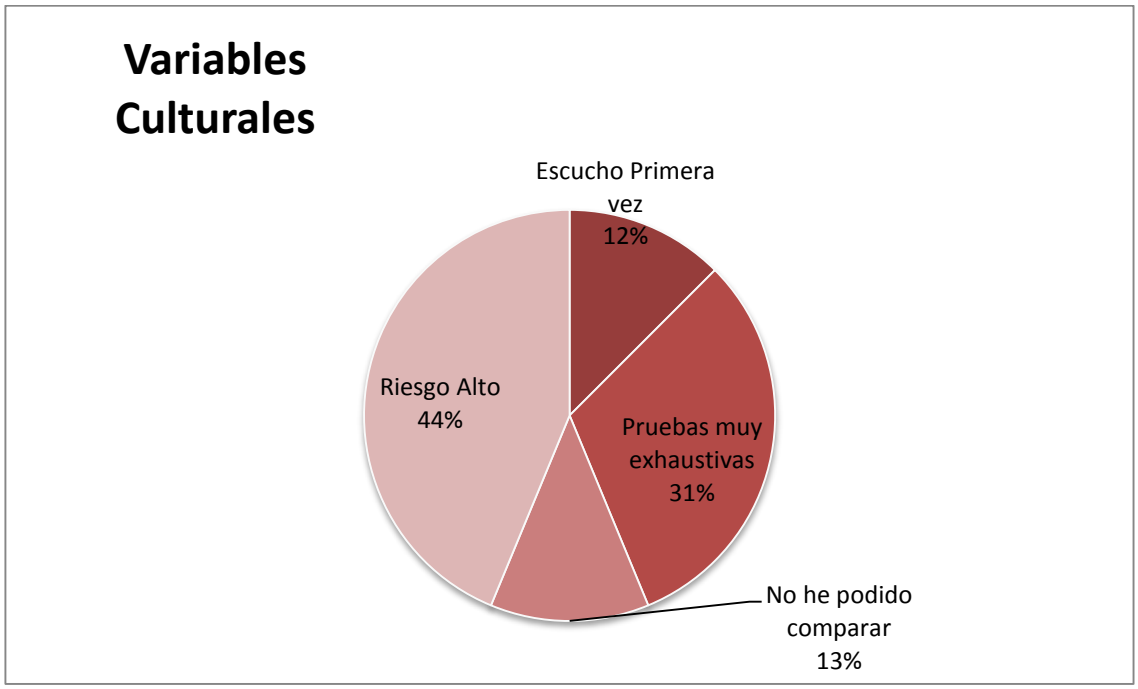


Figura 47: Encuesta: Reconocimiento de las Variables Culturales

El planteamiento de esta pregunta busca el establecer si existen variables exógenas al proyecto, pero específicas en el entorno ecuatoriano, que influyan en la generación de retrasos en los proyectos de software.

El resultado es interesante: más del 70% de los encuestados (Riesgo alto 44%, Pruebas muy exhaustivas 31%) entiende estas variables exógenas como un problema real y de riesgo, que a la larga representa un costo en tiempo y presupuesto a tomarse en cuenta[69].

[69] En los proyectos que he participado, nunca he visto que un planificador coloque tiempo o presupuesto adicional en base a las variables culturales, a pesar que es un tema identificado.

Preguntas abiertas

En la metodología propuesta para la presente investigación se planteó la necesidad de realizar entrevistas a profundidad con diversos actores, los más importantes y representativos del sector, tales como gerentes cercanos o directores de las asociaciones más relevantes.

Sin embargo, durante el diseño y publicación de la encuesta realizada para realizar el levantamiento de datos estadísticos, se decidió cambiar la estrategia del planteamiento de ciertas interrogantes, para que se incorporen las preguntas abiertas planificadas para la entrevista a profundidad, pero dentro de la misma encuesta analítica[70], así evitando realizar entrevistas a profundidad y ampliando el espectro de análisis en los diferentes perfiles de estudio.

Esta decisión se sustentó en los siguientes aspectos:

- Posibilidad de que participen en este tipo de opiniones no solamente los actores más representativos, como gerentes o directores, sino que sea posible el planteamiento de ideas abiertas por parte de los desarrolladores y equipos técnicos. Esto evitaría el sesgo de la visión gerencial y buscaría encontrar ideas nuevas al respecto de los problemas planteados.
- Los gerentes y directores no se restringirían a responder solamente las preguntas abiertas, sino que también llenarían las encuestas, constituyendo que el análisis estadístico cuente con la participación de

[70] En la encuesta, esto corresponde a los casilleros de texto abierto colocados en todas las preguntas, y toda la pregunta 15.

todos los involucrados en un proyecto de software, incluyendo la alta dirección.

- Se tendría un mayor número de puntos de vista sujetos de análisis con ideas a profundidad. Se consideró que es mejor esta decisión que realizar menos de 10 entrevistas dirigidas El resultado sería la recopilación de opiniones de más personas.
- Esta decisión facilitaría también la tabulación y la optimización del tiempo requerido para la revisión y consolidación de las ideas principales.

Las ideas más importantes encontradas en las preguntas abiertas se detallan a continuación:

Pregunta 1

Al respecto de los lenguajes de programación, se señala que hay personas que utilizan otro tipo de herramientas diferentes a las planteadas, dentro de las que se destacan:

- “Genexus”
- “X++”
- “Oracle Apex”

Por el bajo número de opciones no consideradas, puede aseverarse que sí fueron colocadas como opciones las herramientas más utilizadas en el Ecuador.

Al respecto del tipo de negocio en el que han incursionado las empresas, consultado en la primera pregunta, solamente existe una persona que señala que no vende software. Lo más probable es que se trate de una persona que trabaja para una empresa que su giro de negocio no es el software, pero que participa en proyectos de software.

Al respecto del mercado vertical, consultado en la primera pregunta, existen 4 personas que señalaron que faltó detallar una opción específica para el gobierno. Esto es interesante porque nos asegura la diversidad en la selección de la muestra.

Pregunta 6

La pregunta está orientada a determinar si el encuestado considera la medición del tiempo de un proyecto de software como "predecible". Algunos encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "la medición del tiempo se la puede definir cuando conoces dos directrices importantes conocimiento del negocio y conocimiento de la tecnología."
- "Se debe tomar en cuenta el proyecto en sí, el cliente y su necesidad y los factores que pueden influir en base a la experiencia utilizando apoyo de herramientas de desarrollo de proyectos"
- "Líneas de base de la organización"

- "Nada matemático, experiencia del equipo, conocer los fuertes de cada uno, y en mi caso un padaone" [71]
- "Con la información"

Pregunta 7

La pregunta está orientada a determinar si el encuestado considera que la arquitectura de software es "trascendental" en el desarrollo de un proyecto de software. Algunos encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "Utilizamos arquitecturas definidas por el modelo de negocio al que nos orientamos, no utilizamos arquitecturas nuevas"
- "La arquitectura de software no es trascendental pero el uso de una arquitectura eficiente ayudará a la entrega a tiempo de los proyectos"

Pregunta 11

Esta pregunta está enfocada en determinar si las variaciones internas del equipo de trabajo tienen impacto en los retrasos de los proyectos de software. Algunos encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "combinación de la opción 1 y 2" (esto es en la combinación entre contar con un gerente de proyecto y alta documentación)

[71] El término "padaone" viene de la película "StarWars" y es utilizado con cierta frecuencia en el ámbito del desarrollo de software. Se refiere a que un experto trabaja siempre junto a un aprendiz, que va aprendiendo todos los "trucos" hasta que se vuelva un experto también, momento en el cual tiene la responsabilidad de adoptar su propio "padaone" y replicar lo aprendido.

- "Dentro de la gerencia del proyecto, se debe determinar las condiciones de ejecución y planes de contingencia, basados en el presupuesto del mismo. Una práctica base que permite anticiparte a este riesgo es: Documentar, trabajar con estándares de desarrollo y puntos de control de avances. Cualquier otro método que prevenga el riesgo depende de presupuestos, lamentablemente esta etapa no es presupuestada en la definición del proyecto."
- "Lo mejor es contar con un gerente de proyecto, que es el encargado de conocer todos los pormenores de las tareas de su equipo de trabajo. Cuando una persona se va, el líder fácilmente podrá capacitar a otra persona para que continúe con el proceso. Más documentación de lo esencial en la programación y lógica de negocio."

Pregunta 12

Esta pregunta está orientada a determinar si es que la variación de los requerimientos a implementarse en un software, son un motivo central que genera retrasos. Algunos de los encuestados respondieron de la siguiente manera:

- "Una política de control de cambios"
- "SCRUM [72]"

Pregunta 13

Esta pregunta está orientada a determinar si es que la motivación en el personal interno es un factor que influye en los atrasos en los proyectos de

[72] Esta metodología fue definida en la sección de Estado del Arte. Se refiere a una de las metodologías de desarrollo de software "Ágil", que divide el proyecto en múltiples iteraciones pequeñas.

software. Algunos encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "Combinaría la motivación económica, con el constante aprendizaje en paralelo a actividades de distracción o tiempos de descanso cada cierto tiempo o al finalizar determinadas tareas, con la finalidad de incentivar la pronta culminación de una tarea asignada."
- "Juntando el punto 2 (plan de capacitación) más el punto 3 (liberar carga emocional)"
- "Reconocimiento en dinero a medida de la disponibilidad del proyecto, un reconocimiento personal a la labor bien hecha, y comodidades en el ambiente laboral."
- "Los programadores son personas igual a los demás, considero q la opción 1,2,3 irían de la mano porque como persona necesitan de todo tipo de estimulo."
- "Cada logro de algún miembro del equipo debe ser conocido por el grupo, estar al tanto de las necesidades de cada persona y estimularla de acuerdo a su carácter y personalidad. El estimulo económico es bueno pero manteniéndolo controlado. Un equipo de trabajo se motiva cuando tiene un líder que se preocupa de sus necesidades, los defiende y está en constante comunicación con ellos."
- "Una mezcla de las 2 primeras opciones y tratando de definir bien los tiempos de desarrollo" (esto es la motivación económica y el plan de capacitación)

- "combinación de los puntos 1, 2 y 3. En general que se note la preocupación de la gerencia." (esto es la motivación económica, el plan de capacitación y actividades sociales)
- "El modo de motivación al personal debe ser una de las características intrínsecas de una empresa de software. Muchas ocasiones los empleados piensan que su motivación depende de la entrega de cosas que carecen sean estos: beneficios económicos y no económicos, clima laboral, etc. Como en cualquier negocio, el personal es el recurso más valioso y que se debe proteger."
- "Es una mezcla de varias de las alternativas anteriores que deben ser mezcladas de tal manera que logren el resultado y deben ser aplicadas en el momento y circunstancia adecuada"

Pregunta 15

Qué se debería hacer para evitar retrasos en los proyectos de software?

Varios de los encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "Documentación completa inicial revisada y validada por el cliente."
- "No creo que haya una fórmula mágica pero se debería tratar de ser más flexibles en los tiempos planificados."
- "El programador debería realizar lo que está en el documento de Requerimientos técnicos y no debe aceptar cambios en el desarrollo a

menos que se lo realice con un control de cambios para ser nuevamente evaluado tanto en tiempos como en costos."

- "Considerar holguras dentro del proyecto, mantener los objetivos y alcance del proyecto claros."
- "Tener una buena planificación de recursos humanos."
- "Un buen análisis de requerimientos, la arquitectura adecuada y delimitar un alcance"
- "ANALISIS COMPLETO, ESTABLECER UN TIEMPO DE DESARROLLO ACORDE A LA CAPACIDAD DEL EQUIPO, INCLUIR A LOS PROGRAMADORES EN EL ANALISIS, EL JEFE DE DESARROLLO SIEMPRE DEBE ESTAR EN LA IMPLEMENTACION Y CAPACITACION FINALES"
- "buena planificación"
- "Una correcta planificación"
- "Se debería tener un mejor levantamiento de información, e incluso en lo posible tratar de prever por parte de la empresa los posibles cambios en procedimientos en la medida de lo posible"
- "Dedicar mayor tiempo y recursos al levantamiento de requerimientos y análisis del proyecto"
- "Tan solo manejar un proyecto con una metodología base, tanto para la planificación del tiempo como de sus costos. Si se combina con la metodología tradicional para el desarrollo de software se obtendría un mejor resultado."
- "Redefiniendo los ciclos de desarrollo, siendo estos los que determinen los tiempos y no el proyecto en general."

- "SCRUM [73]"
- "Una Planificación de Proyecto bien elaborada"
- "Un buen análisis de requerimientos, definir un buen alcance e identificar riesgos"
- "Se debe detallar exhaustivamente cada etapa y actividad que contempla el desarrollo de software, las personas que intervienen, el cliente final, los cambios internos y externos que pueden suscitarse."
- "Las primeras fases del desarrollo deberán ser al máximo detalle de los requerimientos y funcionalidad que necesita el cliente"
- "Realizar un buen análisis de requerimientos con los usuarios implicados en el manejo de tal o cual proceso, además tener un grupo de trabajo con un ambiente laboral agradable y muy profesional."
- "Establecer una sinergia con el equipo de negocio y técnico del proyecto, considerando todas las variables del mismo (Procesos, personas, tecnología)"
- "Estimar adecuadamente el tiempo de desarrollo además de contar con un colchón de tiempo para riesgos e imprevistos, en base a experiencias previas"
- "Conocer el negocio al que está orientado el proyecto de software. Relevamiento y diseño involucrando usuarios de todos los niveles. Asignar todos los recursos necesarios al proyecto. Evitar ser muy optimista, no tratar de comprometerse con fechas de entrega anticipadas a las generadas en el documento de planificación. "

[73] El término se encuentra explicado en la sección de Estado del Arte. Se refiere a una de las metodologías de desarrollo de software "Agil".

- “repito primero capacitar al equipo de desarrollo tanto en tecnología como en el negocio”
- “Implementar la practica TDD (Desarrollo dirigido por test) en la que se debe realizar pruebas antes de "echar código".”[74]
- “Planificar, documentar, motivar, comunicar.”
- “Realizar un adecuado análisis y diseño. Educar a los clientes sobre el ciclo de vida de un proceso de desarrollo de software.”

Para usted, cuál es el mejor modelo de gestión de una empresa de software / un equipo de software?

Algunos de los encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- “No conozco de modelos de gestión.”
- “Cuáles son los modelos de gestión?”
- “Gerente de Proyecto Líder de Proyecto Arquitecto de soluciones Programador Máster, Programador Junior”
- “Motivar constantemente en cada tarea, generar amigos dentro del grupo”
- “equipo de software”
- “Equipos de trabajo con líderes y tareas definidas”
- “El modelo típico de gestión de una empresa es la que se encuentra administrada por estrategias, donde se identifican causalidades de

[74] En el argot de los desarrolladores de software el término “echar código” se refiere a la etapa de programación.

ejecución y dependencia, en este caso una empresa de software no es ajena a este principio."

- "un equipo con libertad de ejecutar los requerimientos, que deban responder con resultados."
- "SCRUM[75]"
- "Cronograma de trabajo, bajo Planificación"
- "Debe estar estructurado de tal manera que cada persona que se deba hacer cargo de una parte sea especialista en su rama, para que con su experiencia el equipo de software solvente todas las facetas y componentes que intervienen cuando se hacen sistemas informáticos."
- "Un mejor modelo de gestión deberá implicar estándares como el CMMI[76]"
- "Desarrollar ambiente laboral de tal manera que el equipo de trabajo se sienta comprometido y así cumpla con el objetivo de sacar un buen proyecto al menor tiempo posible."
- "Aquel que esté identificados técnica y emotivamente con la relación misional."
- "El modelo clásico o en cascada es el que más he aplicado, a lo mejor por desconocimiento o por costumbre"
- "Si entendí bien la pregunta, seria: Un generador de conocimiento, transferencia del conocimiento y una integración del conocimiento. Esto se logra con una gerencia del proyecto, un equipo de coordinación y los ejecutores."

[75] Se encuentra detallado en la sección de Estado del Arte. Se refiere a una de las metodologías de desarrollo de software "Agil".
[76] Se encuentra detallado en la sección de Estado del Arte. Se refiere a una de las metodologías de desarrollo de software "Agil". Significa Capability Maturity Model Integration (CMMI).

- "el mejor es el que tiene mucha comunicación"
- "El mejor modelo dependerá mucho del tipo de proyecto. Existen modelos probados que pueden ser reutilizados o en su defecto acoplados. No es necesario reinventar la rueda."
- "Enfocado en el factor humano y la calidad del servicio."

Cuál es la mejor manera de hacer rentable / viable el desarrollo de software?

Algunos de los encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "Calculando correctamente el tiempo y recursos que se invertirán en el proyecto."
- "Realizar software genérico que se adapte mejor a las condiciones cambiantes y se pueda seguir mejorando en sucesivas versiones."
- "Definiendo bien el / los requerimientos del cliente."
- "No competir por precio entre la industria, muchas empresas se regalan para conseguir un contrato."
- "Buena planificación."
- "El alquiler del sw[77] seria una de las alternativas a incrementar la rentabilidad de una empresa al dar soporte y recibir el pago correspondiente por dicho servicio"
- "EL REUSO DEL CODIGO"
- "Con calidad buscando la exportación"

[77] Es común utilizar las siglas "sw" para referirse al término "software"

- “Tratar de hacer software con el que se pueda reutilizar código para Varias Empresas”
- “Tener bien claro lo que se quiere y comunicar esto a todo el equipo de desarrollo, las personas que saben el porqué tienen que hacer algo, lo hacen mejor y con mejor actitud que las personas a las que simplemente se les entrega las tareas”
- “La rentabilidad dependerá de la concepción del producto a ofertar, siempre y cuando este asociado a un mercado objetivo, costos de producción y comercialización definidos y ante todo asegurar la calidad del mismo con el ánimo de evitar excesos en el uso de recursos adicionales para su mantenimiento.”
- “la implementación a medida y la distribución masiva de un producto son formas de hacer que un software sea rentable principalmente. Otros modelos de negocios pueden también ser rentables pero en otros escenarios y con otros actores.”
- “margen precio / costo”
- “Cumplir con el cronograma de trabajo”
- “La mejor manera es que tanto el cliente o usuario final esté de acuerdo con el producto que va a recibir, para lo cual se debe detallar cada actividad que va a ser contemplada e involucrarle en la mayor parte del trabajo al cliente que es la parte intelectual del mismo.”
- “evitar retrasos en los proyectos y cargas de trabajo a los desarrolladores por mal levantamiento de los requerimientos.”
- “Estimando los tiempos y los costos de la manera más exacta posible de tal manera que al final se tenga un cliente satisfecho.”

- "Presentando beneficios tangibles"
- "Enfocarse en proyectos que ofrezcan claros beneficios a los clientes
- Tener una línea de especialización. Mantenga el recurso humano. Cumplir con los proyectos dentro de los periodos fijados. No se arriesgue en proyectos bien definidos (recursos y delimitación)"
- "disminuir el tiempo de desarrollo"
- "Otorgando las funciones que se acordaron, cubriendo expectativas y otorgando actualizaciones."
- "Desarrollando productos y servicios de gran calidad, que buscan satisfacer las necesidades de los clientes antes que otra cosa."
- "Aumentando la productividad del equipo con el uso de metodologías reales y efectivas."

Cómo hacer que el software ecuatoriano sea reconocido mundialmente?

Algunos de los encuestados respondieron de la siguiente manera (Se han colocado los textos de forma textual, tal como fue ingresado por los encuestados):

- "Los mejores software deben participar de ferias internacionales para darse a conocer y tener la oportunidad de ser probados."
- "Realizar software de calidad pensando siempre a nivel internacional."
- "Tratando de aplicar las mejores prácticas de programación."
- "Mejorar los controles de QA[78], trabajar en la usabilidad no solo en la funcionalidad"

[78] Hace referencia a las siglas Quality Assurance. Se refiere a la etapa de pruebas y aseguramiento de la calidad de un software.

- “implementando nuevas herramientas para que los demás utilicen y no solo usarlas.”
- “compartir el conocimiento de personas con mas experiencia a las personas que recién ingresan en el mercado laboral, capacitación constante de los desarrolladores”
- “CON CALIDAD Y FACILIDAD DE ENTENDIMIENTO EN SU CONCEPTO”
- “Con el continuo trabajo, con calidad y tratando de mejorar día a día, es solo cuestión de tiempo”
- “Programar un software de calidad que pueda satisfacer la demanda de automatización de varias empresas, tratar de brindar servicios a Empresas Extranjeras que puedan ayudarnos a proyectar nuestra imagen al exterior.”
- “Publicidad, el software ecuatoriano a mi punto de vista es uno de los mejores del mundo (sin tomar en cuenta las ovejas negras del desarrollo) el tema es que el "software ecuatoriano" es contratado por empresas extranjeras y dan la ilusión de que es realizado por otra parte del mundo, ejm. TATA”
- “Todo país debe manejar su nombre como una marca, la misma que demanda infinidad de artificios de comercialización tanto interno como externo. El software ecuatoriano debe embarcarse en esta iniciativa, del mismo modo como el turismo se promociona en ferias a nivel extranjero, nuestro software debería participar sea en ferias o concursos mundiales.”

- "Es reconocido mundialmente. Una forma de ponerlo más arriba de lo que esta es que se puedan implantar factorías[79] de grandes empresas de software (Microsoft, SUN) que inviertan en el país en las zonas francas como las de Quito, Cuenca, Guayaquil. Haciendo a Ecuador un polo de desarrollo de software que genere eventos de mayor categoría que convoque a gentes de distintos lados."
- "QA, QA, QA[80]"
- "Innovador, útil, unificado con herramientas utilizadas mundialmente."
- "Se debería crear asociaciones de empresas de desarrollo de software donde se pueda compartir el producto que se tiene y se pueda impulsar desde un sólo punto todo los sistemas informáticos que existen en el Ecuador y que se pueden exportar."
- "Aplicar estándares internacionales dentro de las empresas los CMMI"
- "Tener un software que cumpla con un alto grado de calidad, y basándose en las normativas propias de cada país."
- "Cumpliendo con los compromisos y estándares de calidad."
- "mejorar la calidad del mismo"
- "Primero ser reconocido localmente. Trabajar con estándares internacionales. Capacitación de personal (ojala certificados). Ser innovador. Trabajar con tecnología de punta."
- "cuando tenga los menores bugs[81] posibles y que sea muy amigable además de una buena documentación"

[79] Con el término factoría se refiere al establecimiento de maquilas de software. Una empresa multinacional utiliza una "zona libre tecnológica" para desarrollar sus productos globales.
[80] Hacer referencia a las siglas de Quality Assurance. Se refiere a la etapa de pruebas y aseguramiento de la calidad del software.
[81] El termino "bugs" se utiliza normalmente en referencia a los errores de programación

- “dejando de utilizar metodologías tradicionales y acoplando los conceptos de extreme programming en lo posible.”
- “Innovando, no hay otro camino.”
- “Con el apoyo gubernamental considerando a la industria del software como una industria estratégica de exportación y crecimiento.”

Profundización sobre los resultados

Para profundizar sobre la información contenida en la base de datos obtenida luego de la ejecución de la encuesta, se realizó diversos cruces de información entre las preguntas, para determinar causalidad o correlación entre las variables utilizadas.

A continuación se presentan los gráficos más representativos de este estudio:

Cruce de información entre la pregunta 1 y 4

Los siguientes gráficos realizan el cruce de información entre las variables de demografía tecnológica y la distribución de tiempo utilizada en sus proyectos, con el fin de demostrar por ejemplo si es que alguna tecnología específica ha implicado un mayor consumo de tiempo de programación o un mayor tiempo de análisis.

En los gráficos presentados a continuación se presentará una comparación entre las dos variables. En el eje horizontal se encuentran las etapas del desarrollo de software, distribuidas de la siguiente manera:

- Definiciones Iniciales = 1
- Análisis y Diseño = 2
- Programación = 3
- Estabilización = 4
- Instalación = 5
- Soporte = 6

Distribución de tiempo por lenguaje

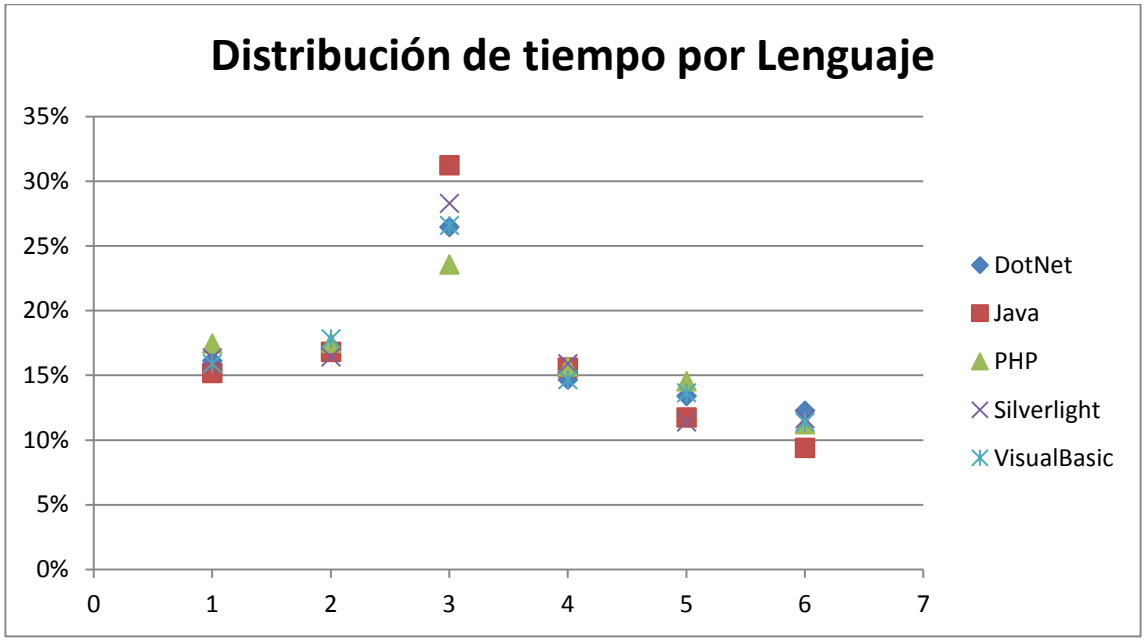


Figura 48: Encuesta: Distribución de tiempo por lenguaje

Se encuentra que para todas las fases del desarrollo del software, las personas que utilizan los diferentes lenguajes de programación distribuyen el tiempo de forma similar, visualizando un indicio de independencia entre las variables de distribución de tiempo y lenguaje de programación. Por ende puede decirse que por razón de un lenguaje de programación, la distribución de tiempo en las diferentes etapas no se ve alterada.

Distribución de tiempo por plataforma

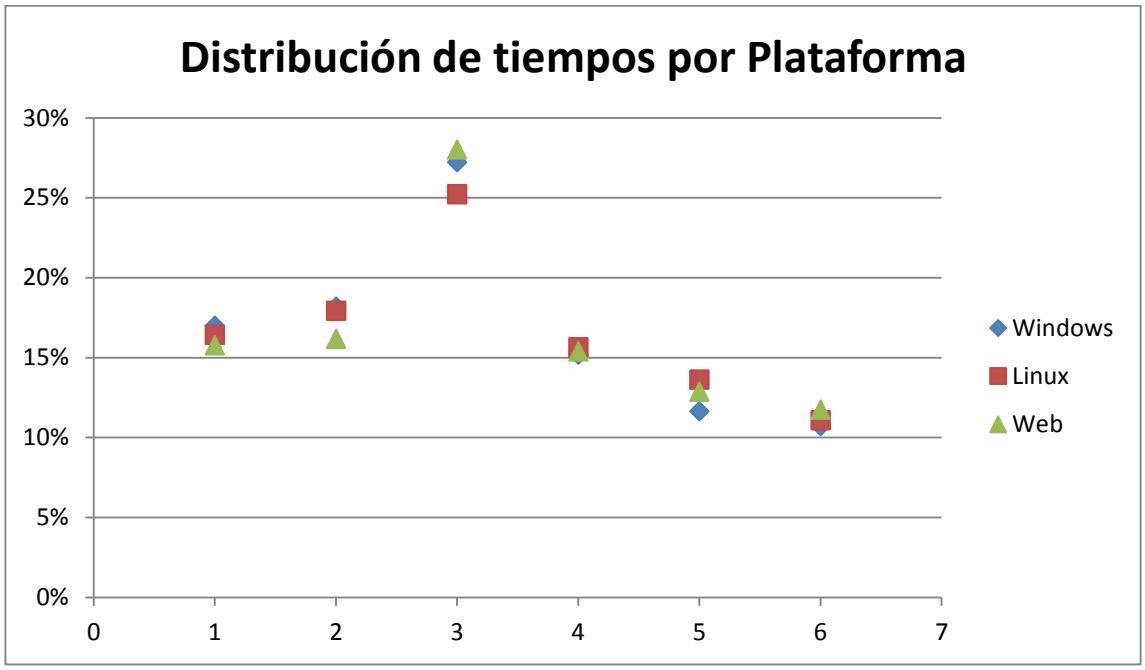


Figura 49: Encuesta: Distribución de tiempo por plataforma

Se encuentra que para todas las fases del desarrollo del software, las personas que utilizan las diferentes plataformas distribuyen el tiempo de forma similar, visualizando un indicio de independencia entre las variables de distribución de tiempo y plataforma. Por ende puede decirse que por razón de la plataforma utilizada, la distribución de tiempo en las diferentes etapas no se ve alterada.

Para demostrar estadísticamente esta relación, se realizó el cálculo del coeficiente de correlación entre los diferentes tipos de plataforma expuestos en la encuesta. El resultado se presenta en la siguiente tabla de datos:

	1	2	3	4	5	6	Coeficiente Correlación
Windows	17%	18%	27%	15%	12%	11%	99%
Linux	16%	18%	25%	16%	14%	11%	97%
Web	16%	16%	28%	15%	13%	12%	

Esto nos indica una alta correlación entre los resultados de las plataformas (promedio = 98%). La presencia de esta correlación alta, significa una baja probabilidad de variación de los porcentajes de distribución de tiempos del proyecto por razón de la variable plataformas.

Distribución de tiempo por modelo de comercialización

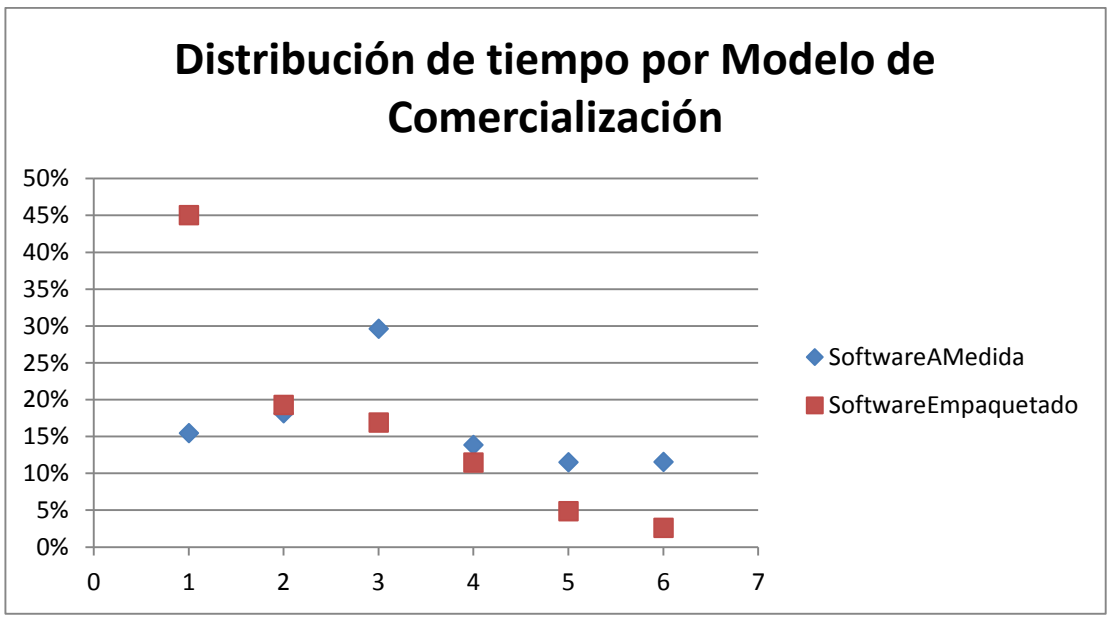


Figura 50: Encuesta: Distribución de tiempo por modelo de comercialización

Se encuentra que las personas que utilizan los modelos de negocio de software a medida utilizan mayor tiempo en las etapas operativas (programación, estabilización, instalación y pruebas), pero el modelo del

software empaquetado toma mucho mayor tiempo en la etapa de definiciones iniciales. Esto se justifica porque el software empaquetado requiere mayor tiempo para lograr la definición inicial del producto.

Distribución de tiempo por la orientación

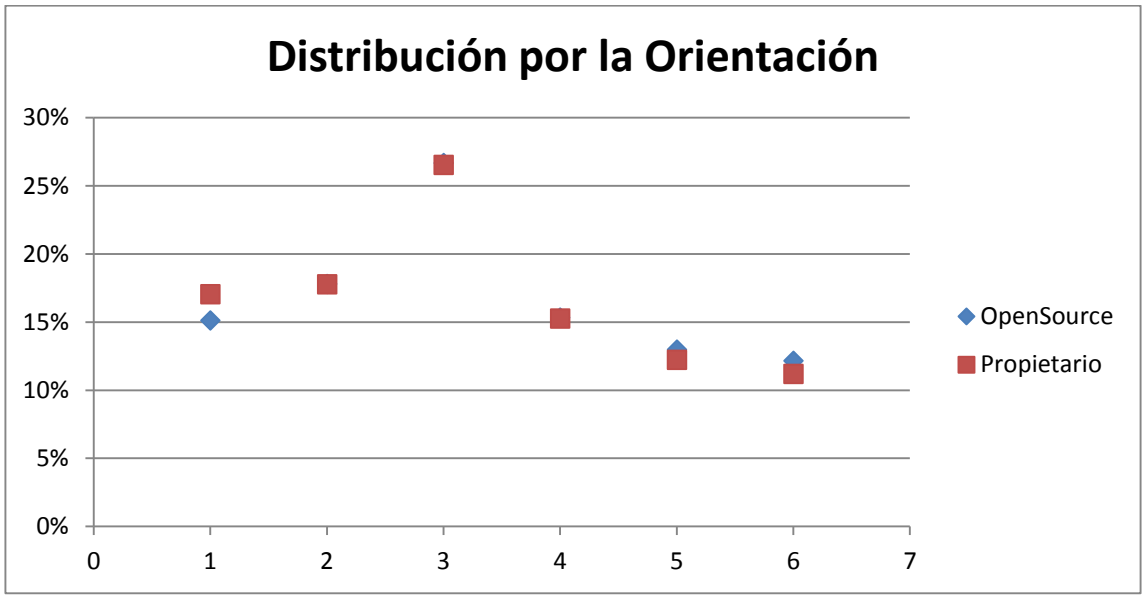


Figura 51: Encuesta: Distribución de tiempo por orientación

Se encuentra que para todas las fases del desarrollo del software, las personas que utilizan software Open Source y software propietario distribuyen el tiempo de forma similar, visualizando un indicio de independencia entre las variables de distribución de tiempo y orientación. Por ende puede decirse que por razón de la orientación, la distribución de tiempo en las diferentes etapas no se ve alterada.

Distribución de tiempo por mercado vertical

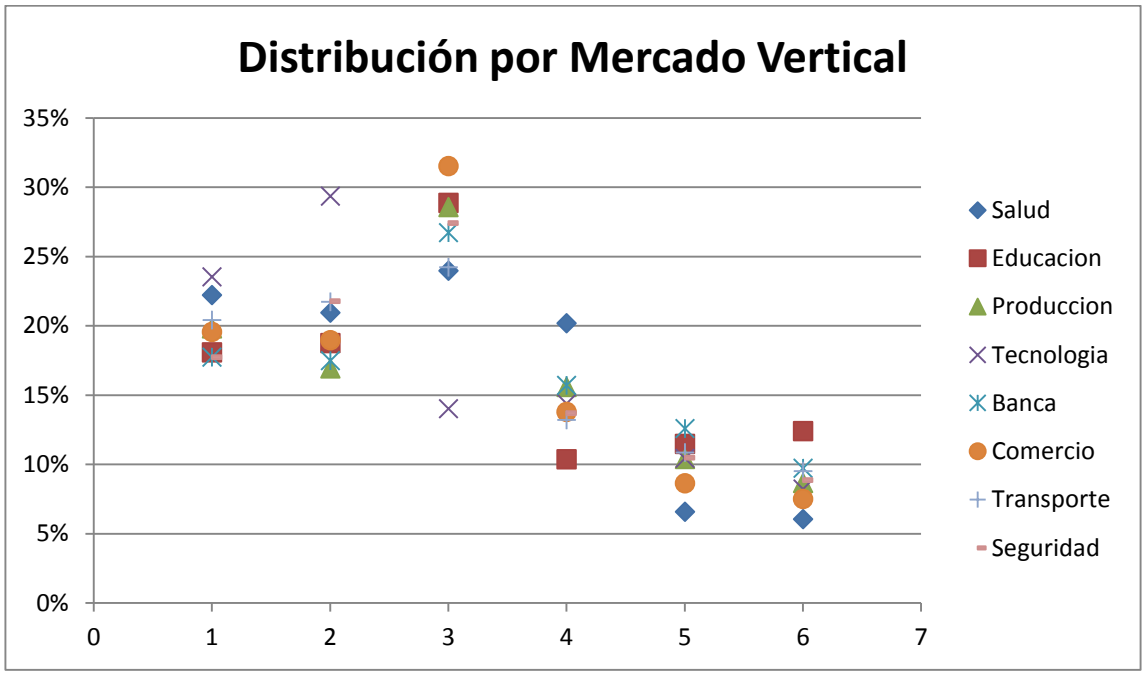


Figura 52: Encuesta: Distribución por Mercado Vertical

Se encuentra que para todas las fases del desarrollo del software, las personas que trabajan para los diversos mercados verticales distribuyen el tiempo de forma similar, visualizando un indicio de independencia entre las variables de distribución de tiempo y distribución del mercado vertical. Por ende puede decirse que por razón del mercado vertical, la distribución de tiempo en las diferentes etapas no se ve alterada.

Para demostrar estadísticamente esta relación, se realizó el cálculo del coeficiente de correlación entre los diferentes tipos de mercado vertical expuestos en la encuesta. El resultado se presenta en la siguiente tabla de datos:

	1	2	3	4	5	6	Coeficiente correlación
Salud	22%	21%	24%	20%	7%	6%	64%
Educación	18%	19%	29%	10%	11%	12%	90%
Producción	20%	17%	29%	16%	10%	9%	35%
Tecnología	24%	29%	14%	14%	10%	8%	33%
Banca	18%	17%	27%	16%	13%	10%	99%
Comercio	20%	19%	32%	14%	9%	8%	93%
Transporte	20%	22%	24%	13%	11%	10%	97%
Seguridad	18%	22%	27%	14%	10%	9%	

Esto nos indica una alta correlación entre los resultados de los mercados verticales (promedio = 73%). La presencia de esta correlación alta, significa una baja probabilidad de variación de los porcentajes de distribución de tiempos del proyecto por razón de la variable mercado vertical.

En resumen, se deduce que no hay una influencia marcada de las variables estudiadas sobre los porcentajes de distribución de tiempo. La conclusión sería que la distribución del tiempo obedece más bien a una característica intrínseca de los proyectos de software, independientemente del lenguaje de programación, plataforma, orientación, modelo de comercialización o mercado vertical.

Existen un par de excepciones en los gráficos que sería interesante analizar. En el gráfico presentado para la variable de distribución por modelo de comercialización, se encuentra que para el software empaquetado se utiliza más tiempo que el promedio en las definiciones iniciales y menos tiempo para la programación.

Otra variación notoria se encuentra en el gráfico que analiza los mercados verticales, en donde se encuentra que el mercado de la tecnología requiere mayor tiempo de análisis, que tiempo de programación.

Cruce de información entre la pregunta 1 y 3

Se realizó el cruce de información entre la pregunta 1 que nos entrega la información de la demografía tecnológica, y la pregunta 3 que nos entrega la cantidad de proyectos que tienen retrasos.

Los gráficos de análisis se encuentran a continuación:

Porcentaje de retraso por Lenguaje de Programación

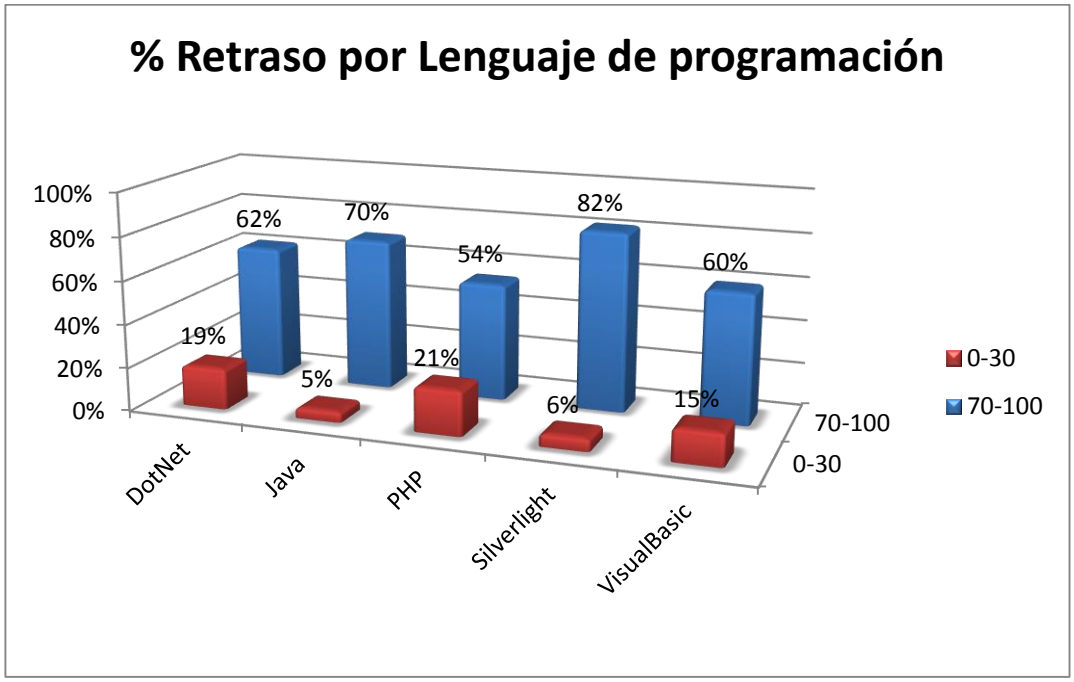


Figura 53: Encuesta: Porcentaje de retraso por Lenguaje de Programación

Las variable “0-30” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan menos del 30% de las veces. Esta variable se refiere a

las personas que tienen muy pocos atrasos en sus proyectos. La variable “70-100” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan en más del 70% de las veces. Esta variable se refiere a las personas que tienen muchos atrasos en sus proyectos. El porcentaje colocado sobre cada una de las barras representa el uso de los lenguajes de programación de las personas que se ubican en la disgregación detallada anteriormente.

Con este gráfico se busca establecer visualmente si es que el uso de un lenguaje de programación específico, explica la existencia de mayores o menores retrasos en los proyectos.

En el gráfico se puede visualizar que no predomina un lenguaje de programación sobre los otros dentro de uno de los grupos, por ende no existe un sustento que lleve a pensar que por ese motivo los proyectos se atrasan.

Aún así, se puede visualizar también la tendencia de que el lenguaje Java y Silverlight tienen mayor posibilidad de generar retrasos en los proyectos que el resto de lenguajes de programación.

Porcentaje de retraso por Plataforma

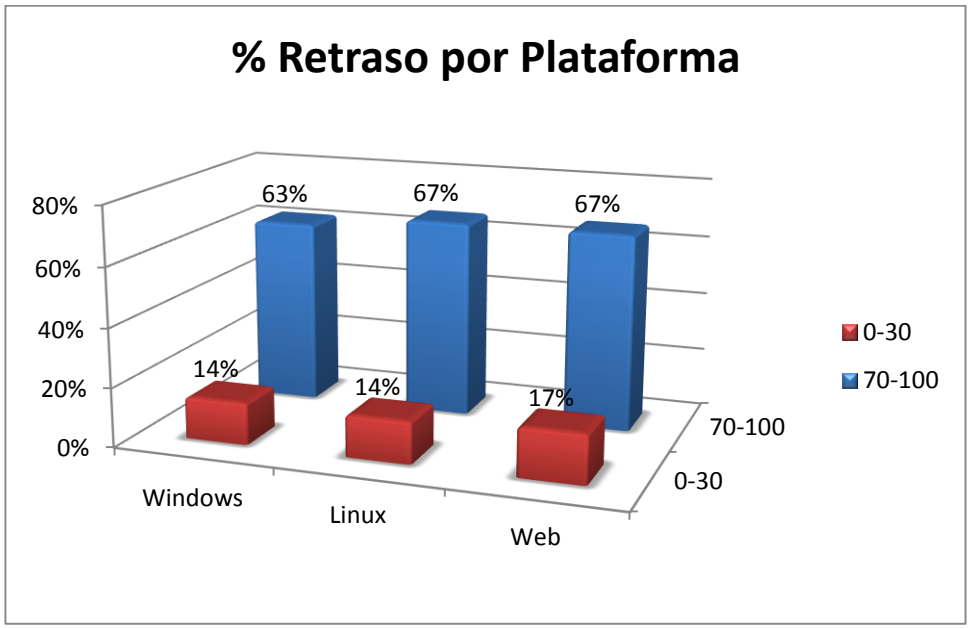


Figura 54: Encuesta: Porcentaje de retraso por Plataforma

Las variable “0-30” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan menos del 30% de las veces. Esta variable se refiere a las personas que tienen muy pocos atrasos en sus proyectos. La variable “70-100” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan en más del 70% de las veces. Esta variable se refiere a las personas que tienen muchos atrasos en sus proyectos. El porcentaje colocado sobre cada una de las barras representa el uso de plataformas que se ubican en la disgregación detallada anteriormente.

Con este gráfico se busca establecer visualmente si es que el uso de una plataforma específica, explica la existencia de mayores o menores retrasos en los proyectos.

En el gráfico se puede visualizar que no predomina una plataforma sobre las otras dentro de uno de los grupos, por ende no existe un sustento que lleve a pensar que por ese motivo los proyectos se atrasan.

Porcentaje de Retraso por Modelo de Comercialización

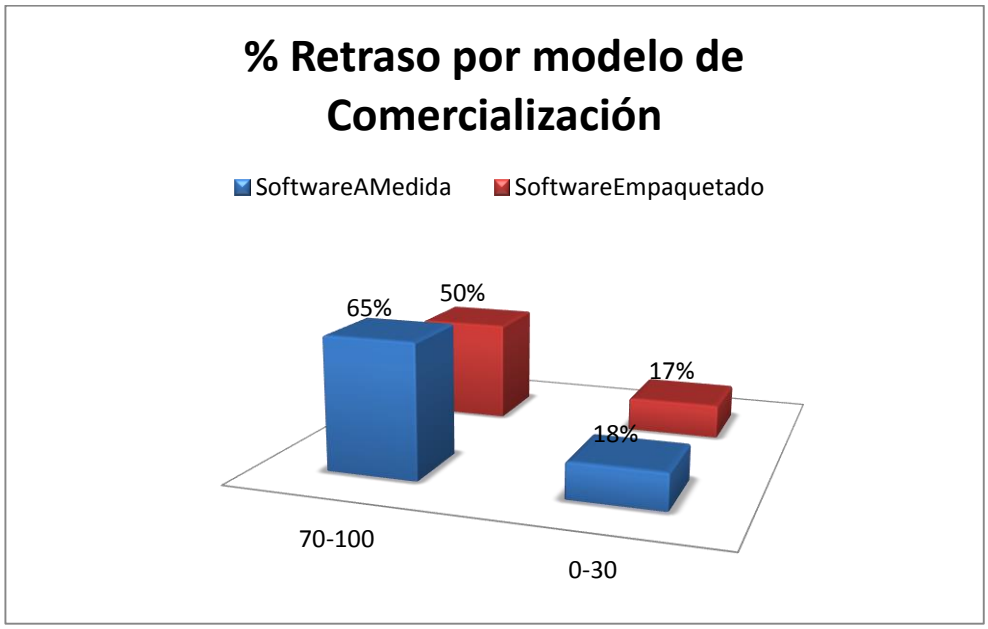


Figura 55: Encuesta: Porcentaje de retraso por Modelo de Comercialización

Las variable “0-30” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan menos del 30% de las veces. Esta variable se refiere a las personas que tienen muy pocos atrasos en sus proyectos. La variable “70-100” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan en más del 70% de las veces. Esta variable se refiere a las personas que tienen muchos atrasos en sus proyectos. El porcentaje colocado sobre cada una de las barras representa el uso de los modelos de comercialización de las personas que se ubican en la disgregación detallada anteriormente.

Con este gráfico se busca establecer visualmente si es que el uso de un modelo de comercialización específico, explica la existencia de mayores o menores retrasos en los proyectos.

En el gráfico se puede visualizar que no predomina un modelo de comercialización sobre los otros dentro de uno de los grupos, por ende no existe un sustento que lleve a pensar que por ese motivo los proyectos se atrasan.

Porcentaje de retraso por Orientación

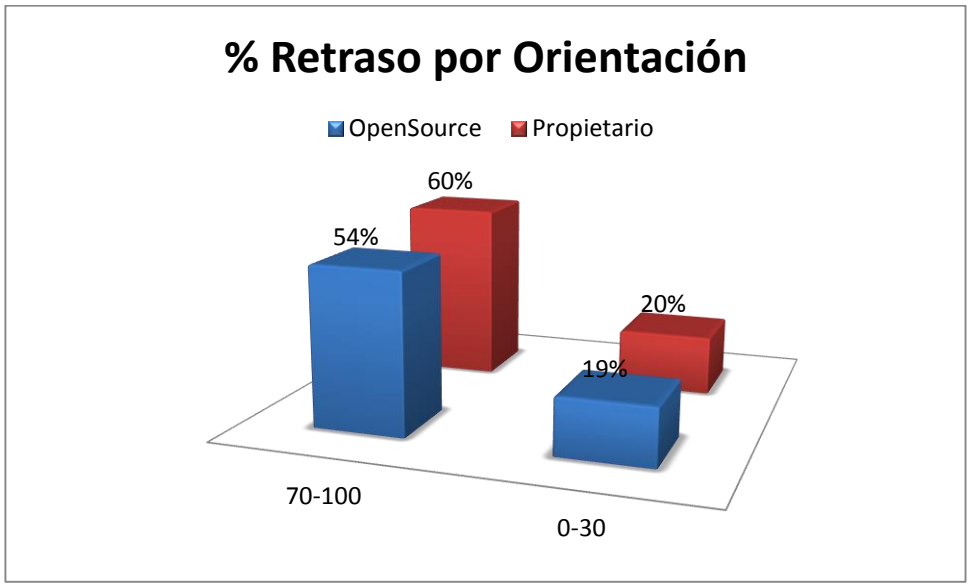


Figura 56: Encuesta: Porcentaje de retraso por Orientación

Las variable “0-30” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan menos del 30% de las veces. Esta variable se refiere a las personas que tienen muy pocos atrasos en sus proyectos. La variable “70-100” se refiere a las personas que en la encuesta dijeron que sus proyectos se

retrasan en más del 70% de las veces. Esta variable se refiere a las personas que tienen muchos atrasos en sus proyectos. El porcentaje colocado sobre cada una de las barras representa la orientación de las personas que se ubican en la disgregación detallada anteriormente.

Con este gráfico se busca establecer visualmente si es que el uso de una orientación específica, explica la existencia de mayores o menores retrasos en los proyectos.

En el gráfico se puede visualizar que no predomina una orientación sobre las otras dentro de uno de los grupos, por ende no existe un sustento que lleve a pensar que por ese motivo los proyectos se atrasan.

Porcentaje de retraso por Mercado Vertical

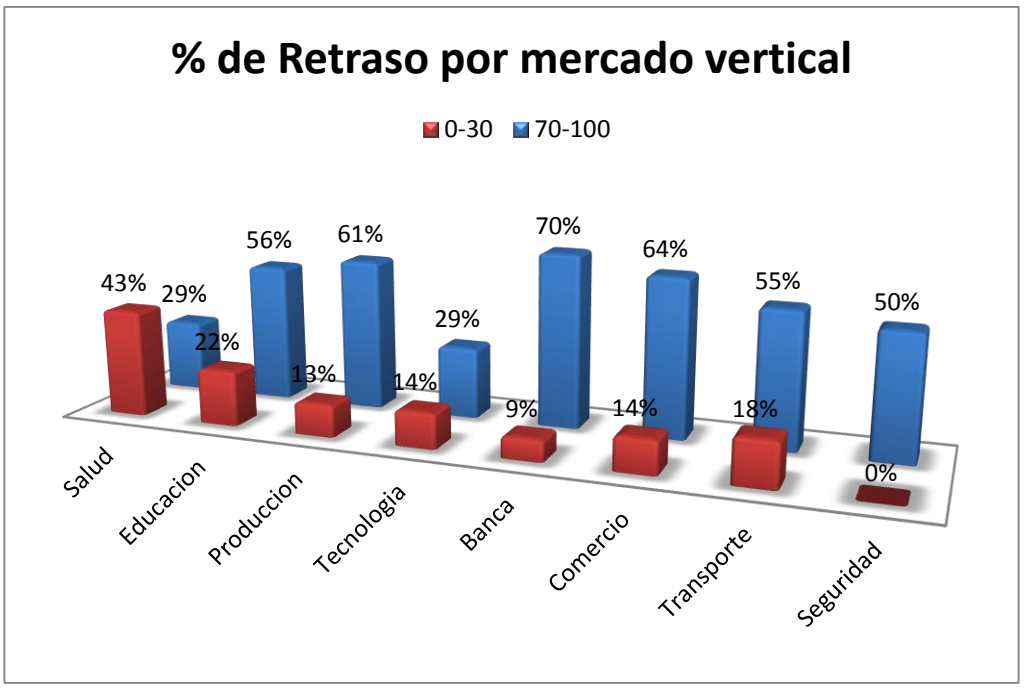


Figura 57: Encuesta: Porcentaje de retraso por Mercado Vertical

Las variable “0-30” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan menos del 30% de las veces. Esta variable se refiere a las personas que tienen muy pocos atrasos en sus proyectos. La variable “70-100” se refiere a las personas que en la encuesta dijeron que sus proyectos se retrasan en más del 70% de las veces. Esta variable se refiere a las personas que tienen muchos atrasos en sus proyectos. El porcentaje colocado sobre cada una de las barras representa el mercado vertical de las personas que se ubican en la disgregación detallada anteriormente.

Con este gráfico se busca establecer visualmente si es que el uso de un mercado vertical específico, explica la existencia de mayores o menores retrasos en los proyectos.

En resumen, es posible visualizar que para todas las variables analizadas se presenta un promedio similar, alrededor del 60% de sus proyectos tiene retraso sin dependencia de la distribución demográfica estudiada.

Existe solamente un caso excepcional cuando se analizó el gráfico por mercado vertical, se encontró que en el sector de la salud había una tendencia inversa. Esto puede ser el resultado de que apenas hubo 3 encuestados que se ubicaron dentro de esta categoría. Sería interesante realizar un análisis posterior a este estudio para profundizar en esta variable específica.

Cruce de información entre la pregunta 8 y 3

Se realizó la comparación entre la pregunta 8 que entrega información al respecto de los métodos utilizados para la valoración de los tiempos que toma un proyecto de desarrollo de software, y la pregunta 3 que entrega información al respecto de los porcentajes de retraso de los proyectos.

A continuación se presenta el gráfico de este análisis:

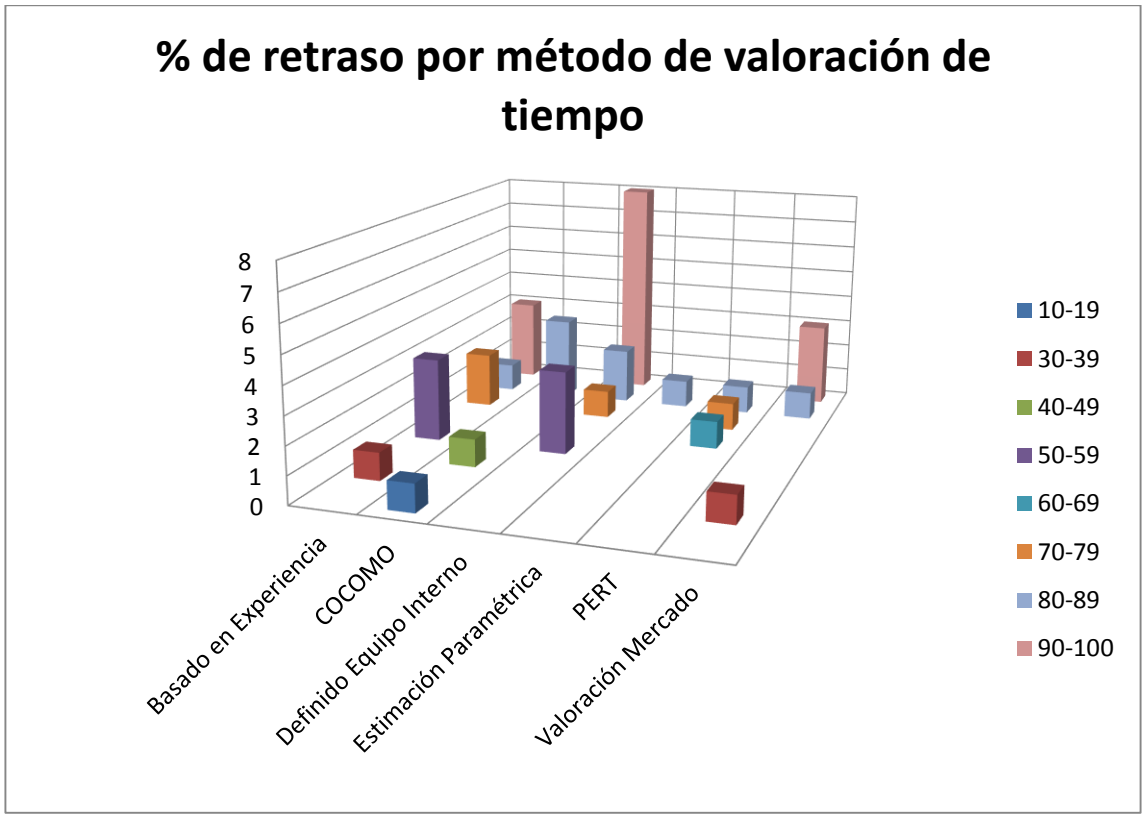


Figura 58: Encuesta: Porcentaje de método de valoración de tiempo

En este gráfico es posible encontrar algunas conclusiones importantes:

- La mayoría de los atrasos fue motivado cuando se utilizó el método de definición con el equipo interno de trabajo.
- Los proyectos que tuvieron los más bajos retrasos, utilizaron el método matemático llamado COCOMO. Sin embargo su proporcionalidad con

respecto al resto de los proyectos no es muy significativa. Sería importante realizar posteriormente un estudio especializado que nos permita comprender si es que la utilización de dicho método matemático hace que la tendencia al retraso se disminuya. Esto también es motivado por la diferencia de opiniones, ya que otros encuestados que también utilizan método COCOMO, generaron retrasos mayores.

- Si sumamos el porcentaje de los métodos “no matemáticos”, podemos encontrar que se encuentra ahí la mayoría de las respuestas de los encuestados.
- Luego de analizar este gráfico, es posible señalar que no hay una variable claramente influyente que nos permita aseverar que por el uso de un método en específico se generaron menos retrasos.

Capítulo 5: Análisis sobre los resultados

Luego de haber estudiado el resultado de las encuestas, tanto como las respuestas obtenidas en las preguntas abiertas, e información adicional de la presente investigación, se considera importante presentar las siguientes reflexiones:

1. Luego de revisar los gráficos estadísticos, ha sido posible confirmar la hipótesis de que el problema de los retrasos en tiempo y presupuesto para los proyectos de software son reales, tangibles, recurrentes y actualmente representan un problema importante en la industria ecuatoriana. Esto significa que en general existe una alta posibilidad de que un proyecto de software en el país tenga retrasos, se ha calculado el 57% de probabilidad de que un proyecto tenga retrasos según la información de las encuestas[82].
2. Lastimosamente no es posible asegurar que los retrasos son motivados por un aspecto o condición específica que nos permita proponer mejoras directas y obtener cambios inmediatos y significativos. Luego de analizar los resultados, puede decirse que se trata de un problema en donde influyen muchas variables a la vez, ya sean internas o externas al proyecto, que corresponden más bien al escenario de ejecución del software y de las prácticas que se han utilizado comúnmente.

[82] Este valor se encuentra justificado en el título "Análisis estadístico de la encuesta", en la Pregunta 3. La probabilidad estimada se calculó utilizando una curva normal formada por los datos recopilados en la encuesta..

3. La mayoría de empresas distribuyen proporcionalmente la misma cantidad de tiempo para cada una de las etapas del software. Lo que ha parecido un hallazgo importante es que dentro de estas distribuciones, en promedio las empresas utilizan el 40% del tiempo entre definiciones iniciales y análisis y diseño, y cuando se preguntó sobre los motivos del retraso, más del 40% de los encuestados dijo que se trataba de temas como "desconocimiento del negocio", "errores en la planificación" o "errores en la definición del alcance": Esto permite ilustrar una contradicción clave, porque por un lado puede decirse que las empresas invierten bastante del tiempo disponible en planificar, pero cuando los proyectos se atrasan, se señala que es por motivo de errores justamente en esta etapa. La conclusión es que existe una deficiencia teórica / práctica en la metodología utilizada en la etapa de definiciones iniciales y análisis y diseño, que está ocasionando que se pierda mucho tiempo en dicha etapa sin lograr resultados efectivos, generando retrasos a todo el proyecto.
4. Un hallazgo encontrado luego de la investigación es al respecto de la profundidad e impacto del problema, porque al iniciar la investigación se tenía una idea clara de que había retrasos, pero en ese momento no era posible cuantificar cuál era la magnitud de los atrasos en función el tiempo planificado. Luego del estudio pudo confirmarse una realidad preocupante, ya que la diferencia entre lo planificado y lo ejecutado se ubica en alrededor del 39%[83]. Esto significa que los problemas de planificación detectados están consumiendo las utilidades de las

[83] El detalle del cálculo del valor expuesto puede encontrarse en el título "Análisis Estadístico de los Resultados", en la Pregunta 5.

empresas, y/u ocasionando constantes renegociaciones de plazo con el cliente final[84]. Casi podría asegurar sin hacer un estudio formal, que si tomamos como muestra a un grupo de empresas medianas y grandes que han desarrollado software internamente o han contratado empresas ecuatorianas para el desarrollo a medida, ya tienen desconfianza de los ofrecimientos de la industria. Esta "mala fama" también probablemente ocasiona inconvenientes cuando una empresa nacional intenta incursionar en mercados extranjeros.

5. La mayoría de mediciones de tiempo y presupuesto realizadas en la industria ecuatoriana del software, son realizadas en función de la experiencia y percepción del mercado. Esta confianza que se tiene en el "ojo" del planificador implica riesgos, sin embargo, los modelos matemáticos propuestos por las grandes casas comerciales y entidades de estandarización[85] tampoco aseguran una medición precisa para determinar estas variables. El fundamento de estas argumentaciones es el hecho de que el software no es un producto tradicional, sino que está en constante evolución, por lo que es muy complicado buscar la planificación exacta hoy, de algo que no se sabe exactamente como será y variará en el futuro. Esto justificaría el hecho de que las empresas invierten mucho tiempo tratando de dilucidar como debe hacerse el software en las etapas de definiciones iniciales y análisis, sin llegar a resultados fiables. No obstante, he visto que esta premisa no es tomada

[84] Una negociación de plazo con el cliente normalmente es dada por acuerdo entre las partes, sin embargo, no es común que se encuentren negociaciones de ampliación de plazo que signifiquen ampliación del presupuesto. El argumento utilizado por los clientes finales es que han contratado la provisión de un producto a un precio cotizado, y eso se debería respetar. Esta "puja" con los clientes hace que la gerencia de este tipo de proyectos, en algunos casos sea muy desgastante.

[85] Dentro de este tipo de modelos se encuentra COCOMO, PERT, PROBE, Estimación Paramétrica, Putnam y otros que fueron expuestos en el capitulo llamado "Estado del Arte".

en cuenta en el momento de planificar un proyecto de software, y se le propone al cliente un cronograma exacto de ejecución. En este sentido, se ha visto una interesante aproximación a la solución del problema en las metodologías denominadas "ágiles", que proponen que el software variará constantemente su alcance mientras se va desarrollando, haciendo iteraciones constantes para ir ajustando el rumbo de la planificación. Por este motivo esta metodología está empezando a ser utilizada también en proyectos no tecnológicos[86]. Lastimosamente estas metodologías ágiles van en contra de las formalidades establecidas en un contrato común de provisión de software, en donde se exige la definición de una fecha exacta de entrega, y si se sobrepasa dicha fecha existen multas aplicables. Tal vez para solucionar este escenario de forma óptima, se requeriría que existan modificaciones legales fundamentales que especialicen su reglamentación específicamente para los proyectos de software, lo cual suena improbable que suceda.

6. Cuando una persona se retira de un equipo de trabajo, se lleva consigo mucho del conocimiento aprendido durante la ejecución del proyecto. Esto sucede no solamente en los proyectos de software, sino en diversas industrias. Se encuentra que las empresas de software utilizan tres estrategias principales para mitigar este problema: la contratación de un gerente de proyecto, generación de una alta documentación, y establecimiento de personal de respaldo. Esto permite concluir que las empresas actualmente están invirtiendo significativos recursos de tiempo

[86] Su modelo conceptual está siendo utilizado en diversas áreas diferentes al desarrollo de software, fundamentalmente en los departamentos de Investigación y Desarrollo de las empresas, y otras áreas que necesitan un proceso de evolución y mejoramiento constante.

y económicos para mitigar este problema, pese a eso, siguen existiendo retrasos.

7. Es interesante encontrar que para la gran mayoría de los encuestados, el contar con un clima laboral adecuado tiene una alta relevancia en relación con la productividad de los empleados. Esta aseveración propone entonces la interrogación de ¿cómo establecer un clima laboral adecuado?. La experiencia personal indica que para muchos empleados, un adecuado clima laboral significa que puedan generar amistad con sus compañeros, poder conversar de temas diversos durante el trabajo, poder compartir temas personales, de reunirse para realizar actividades sociales y otras, que en general podrían considerarse distractores del trabajo directo. Esto permite encontrar un punto de análisis profundo, pues es de notar que cuando este tipo de actividades se realizan internamente en el equipo de trabajo, también mejoran los niveles de comunicación y coordinación, mejora la efectividad de las horas trabajadas y el apoyo mutuo para lograr los objetivos. Entonces será decisión de la gerencia el determinar la cantidad de tiempo dedicada a mejorar el clima laboral para disponer de un ambiente de trabajo sano y amigable, pero lo suficientemente limitado como para que los empleados no distraigan sus actividades encomendadas.
8. Uno de los hallazgos destacados en el presente estudio es el referente a la motivación al personal, pues para muchas personas en el área se hubiera considerado que la motivación económica sería lo más importante para asegurar una productividad y compromiso adecuado, sin

embargo se demuestra que este medio de motivación sin dejar de ser importante no es el más relevante. El que se ubicó en el primer lugar es el establecimiento de un plan de capacitación. Esto confirma el hecho de que los equipos de desarrollo de software son personas que se motivan cuando están al día con la tecnología, cuando encuentran mejores formas de hacer su trabajo, cuando tienen dinamismo en su profesión, cuando aprovechan, adaptan y retroalimentan ideas de otros profesionales, cuando mejoran su ego profesional[87]. Probablemente este es el motivo por el cual se ha encontrado que los foros profesionales publicados en el Internet relacionados con la tecnología, están constantemente en movimiento.

9. Durante la gestión de los proyectos de software, uno de los asuntos que ha sido más difícil de controlar, es el referente a la variación de requerimientos durante el proyecto. Esto sustenta la hipótesis de que el software es un proceso de mejora constante, antes que un producto, ya que en un proyecto de software se dan este tipo de cambios de forma impredecible y recurrente. Muchos de los participantes del estudio señalaron que el llevar documentación formal les permitía protegerse de este problema, sin embargo, considero muy difícil desarrollar un nivel de documentación tan detallado como para que se pueda demostrar efectivamente que se trata de una variación, y si así se lo lograra, tomaría un tiempo bastante alto su elaboración. Además a ojos del cliente final, el hecho de que se lleve una documentación estricta de

[87] El tema del ego profesional es un punto que se pudo evaluar recientemente con un colaborador, que ha significado un punto de quiebre en la forma de cómo hacer motivación interna la empresa, y que se refiere a motivar a los empleados haciéndoles sentir importantes y destacados en la "sociedad tecnológica".

cambios es sinónimo de inflexibilidad y por ende de baja calidad en la atención a los requerimientos del usuario. En la encuesta se encontró que más del 40%[88] de las empresas optan por llevar una documentación moderada y flexibilizar un poco la atención al usuario, protegiéndose con documentos de control de cambios más sencillos, aceptando ciertas variaciones como parte del proyecto y otras pidiendo a los clientes que acepten una variación que modifique tiempo y presupuesto (aunque para que se acepte, a veces es necesario procesos de negociación desgastantes). El problema de esta perspectiva es que los cambios no son absolutamente documentados y su magnitud se basa en la percepción de los planificadores, que en algunos casos pueden asumir dentro del alcance del proyecto requerimientos parecen sencillos, pero a nivel de programación requieren la modificación de bases estructurales. Uno de los puntos más preocupantes dentro de este análisis fue el hecho de que el 27%[89] de los encuestados declara abiertamente que no tiene una forma adecuada para controlar este problema, presentando una deficiencia teórica / práctica evidente.

10. Una de las inquietudes planteadas al inicio del proyecto señalaba que en el Ecuador el desarrollo de software tiene elementos singulares y característicos que lo diferencian con los otros países. Luego de la investigación se encontró que existía consciencia de este problema en el 75% de los encuestados[90], señalando en su mayoría que era necesario

[88] Es posible ampliar el análisis de este valor, refiriéndose al título "Análisis estadístico de los resultados", en la Pregunta 12.
[89] Es posible ampliar el análisis de este valor, refiriéndose al título "Análisis estadístico de los resultados", en la Pregunta 12.
[90] Es posible ampliar el análisis de este valor, refiriéndose al título "Análisis estadístico de los resultados", en la Pregunta 14. El valor representa la suma de la opción "Alto Riesgo" y "Pruebas muy exhaustivas".

invertir tiempo adicional del proyecto para ejecutar procesos de pruebas más intensos y aún así representaba un alto riesgo durante las entregas de los proyectos.

PARTE III: Análisis de Escenarios y Buenas Prácticas

"Cuando se sabe de grandes, complejas y costosas iniciativas auspiciadas por el gobierno que no cumplen con las expectativas, la culpa casi siempre la tiene el cronograma "apurado". Si la institución pública no hubiese sido tan exigente con los contratistas, si sólo los desarrolladores hubiesen tenido más tiempo para refinar el diseño, si sólo se hubiesen incluido más meses o años en el programa de producción, la tecnología hubiese funcionado a la perfección. Pero nada de esto es verdad."

"Incluso si los patrocinadores gubernamentales pueden dar a los contratistas todo el tiempo y dinero del mundo para completar un detallado análisis de requerimientos y un diseño de proyectos perfecto, es probable que el producto terminado igual tenga fallas."

"El llamado "diseño big- bang", donde los desarrolladores realizan un enorme esfuerzo para que hasta el último detalle esté correcto, no puede entregar un producto que funcione completamente sin errores porque los seres humanos, no importa cuán brillantes sean, no son capaces de prever todos los problemas que pueden surgir cuando se trabaja con una tecnología compleja."

"Lo que se necesita en su lugar es un proceso iterativo de desarrollo en que se construye un prototipo o incluso un producto terminado, y después se prueba en terreno y se descubre cuáles son sus debilidades, la mayoría de las cuales no se podrían haber previsto en una pizarra o en una simulación 3D. En la siguiente iteración, se mejora el producto. Después se realiza otra ronda y otra y otra, hasta que finalmente se llega a algo que funciona. Mientras más compleja la tecnología, más iterativo debería ser el proceso de desarrollo" (Moe, 2011)

Este texto fue desarrollado por Gary Moe en respuesta a un caso de estudio de tipo administrativo / gerencial que analiza la factibilidad de invertir en un proyecto promocionado por la Administración Aeronáutica Canadiense, que involucraba entre otras cosas, el desarrollo de software y tecnología para una estación espacial. Este texto nos demuestra el sinnúmero de dificultades y contradicciones que se encuentran durante la planificación de un proyecto de ingeniería. Si una agencia espacial presenta dichos problemas, a pesar de sus amplios presupuestos y participación de expertos reconocidos mundialmente, qué podríamos decir de proyectos como los desarrollados por las empresas ecuatorianas que por lo general tienen muchas más limitaciones.

La tercera parte de la presente investigación tiene como objetivo proporcionar diversos planteamientos de solución a manera de "buenas prácticas" sobre los problemas detectados durante el presente estudio, utilizando la experiencia sobre este tipo de escenarios, los conocimientos y técnicas recopiladas en la investigación directa, y múltiple documentación gerencial publicada en revistas y el Internet.

Para lograr un mayor impacto en la presentación de las "buenas prácticas" y para facilitar su difusión, se utilizó un lenguaje jovial y directo encaminado al gerente de proyectos de software, pues es quién tiene la responsabilidad de establecer los tiempos y presupuestos, y controlar la ejecución. Adicionalmente se ha segmentado a cada uno de los problemas con un caso de estudio ficticio, que sirva como escenario base para realizar el tratamiento de los problemas planteados y sus posibles soluciones.

Capítulo 6: Al respecto de la estimación del proyecto y planificación inicial

Escenario

Tú formas parte de una empresa que desarrolla software a medida. Cierto día don Luis Zapatero, dueño de una empresa de fabricación de zapatos deportivos, toma contacto con tu empresa para que le asesoren en la cotización de un software a medida para el control administrativo de su empresa. El cliente se explica de la siguiente manera durante la reunión de levantamiento de información inicial:

"Requiero un módulo contable, un módulo de impuestos y un módulo específico para el control de la producción de zapatos deportivos.

La contabilidad es la básica y comúnmente utilizada en cualquier tipo de empresa, por lo que 'no tiene nada fuera de lo común', eso no necesito explicarte con mayor detalle, 'usted debe conocer esto'...

El módulo de impuestos está relacionado con la contabilidad, pero sólo para sacar de los datos, y de igual manera es la comúnmente utilizada por empresas, así que no creo que vayas a tener problemas…

Eso sí, es importante que el sistema me maneje los anexos transaccionales y las declaraciones electrónicas que ahora está pidiendo el SRI. De todas maneras esto de las declaraciones lo tiene cualquiera de "esos programas que venden en la calle"…

El módulo de producción es lo único 'especial' y por lo que realmente quiero un nuevo sistema integrado, lo que necesito es controlar los inventarios de materia prima y producto terminado, y contabilizar automáticamente los movimientos.

Ten en cuenta que la idea es hacer algo básico. No te compliques mucho en tu análisis, más bien puede decirse que es algo sencillo y no debe ser muy costoso. Fíjate que hasta ahora mucho de esto lo llevo bien en Excel nomás, pero ya quiero formalizar en un sistema propio.

Ayer también vinieron unos jóvenes recién graduados de sistemas, que me dijeron que me sacaban el sistema súper pronto, pero por eso me contacto contigo que me han comentado que tu empresa tiene experiencia, para poder comparar.

Eso sí, ayúdame con la cotización urgente, que quiero arrancar lo más pronto, porque necesito tener este sistema funcionando antes de Junio."

Luego de esta explicación y otras explicaciones complementarias, le ofreces entregar una cotización.

Una vez presentado el caso, se plantean las siguientes interrogantes:

- Qué consideraciones tomarías en cuenta para poder elaborar la cotización solicitada?
- Cómo definirías el tiempo y el precio a colocar en la propuesta?
- Cuánto tiempo te tomarías para realizar el estudio del proyecto?

Estudio de Buenas Prácticas

LECCIÓN 1

Es importante tener en cuenta que si anteriormente no has tenido la oportunidad de obtener experiencia en un sistema similar en el área administrativa / financiera, y específicamente al respecto de módulos contables y de impuestos, te será complicado entender la funcionalidad de los módulos que requiere el cliente, y menos aún realizar una cotización de forma directa.

En la redacción del caso, el cliente no está dispuesto explicar temas que para él son básicos, y considera que deberías saber esos temas como "conocimiento general", pero un ingeniero en sistemas como cualquier persona difícilmente tiene la oportunidad de conocer a detalle la gran cantidad de sistemas existentes en el mercado[91] y menos aún los conceptos utilizados en cada giro del negocio. Lastimosamente los clientes a veces asumen esto, y de aquí parten en algunos casos los errores en las definiciones del alcance del proyecto.

Así sea que el cliente tenga todas las intenciones de explicarnos todos los requerimientos y las condiciones de su negocio, he aprendido que un cliente nunca nos entrega toda la información necesaria, generalmente originado en su propio desconocimiento[92] o por un sesgo situacional, y por ende no debemos basarnos solamente en ello para hacer nuestras proyecciones.

[91] Es novedoso encontrar que un buen porcentaje de ingenieros de sistemas a veces tienen grandes dificultades para usar Excel o PowerPoint, que para muchos usuarios son las herramientas más básicas que existen.

[92] Luego de varios años de realizar análisis de sistemas, he podido percibir que prácticamente ninguno de los clientes que me han entregado requerimientos sabe exactamente lo que quiere. Mientras se va profundizando, se va encontrando nuevas condiciones y el analista planteando una serie de reflexiones

Siempre es necesario investigar en libros o Internet los sistemas similares, las metodologías utilizadas, las prácticas de negocio relacionadas y toda la información que nos permita conocer con mayor profundidad el concepto negocio a ser implementado, ya sea dentro del país o inclusive en el extranjero.

Mientras más se conoce del giro del negocio, más podemos entender y comparar entre lo que el cliente "quiere" y lo que el cliente "necesita"[93], y en base a ello proponer optimizaciones tecnológicas y también metodologías y procesos no tecnológicos[94].

Es por ello que contrario a la teoría de mercado comúnmente utilizada, puede decirse que en la planificación de un software "el cliente NO SIEMPRE TIENE LA RAZÓN", y por ende un software no debe programarse textualmente como el cliente lo pide, aunque luego hay que convencerle de las nuevas ideas durante las exposiciones y negociaciones.

Este ejercicio es muy enriquecedor y motivante para el Ingeniero en sistemas, porque parte de su trabajo regular es el aprender cada día nuevas cosas de negocios diferentes, y dichos aprendizajes le dan experiencia para tratar cada vez escenarios más complejos.

Lastimosamente el tiempo disponible para hacer una cotización o planificación de software, es normalmente muy breve, hablamos de lapsos promedio de una

---

que antes no se habían planteado en el negocio. Entiendo que esto está motivado en el hecho de que el manejo de la "complejidad" para cualquier humano difícilmente es lo suficientemente completa y descriptiva como lo es la programación de un sistema.

[93] Nunca olvidaré la frase dicha por uno de mis profesores de la Universidad que decía que el desarrollar software no significa el trasladar el "caos en papel" al "caos informático", siempre deben estudiar los procesos previamente y buscar optimizarlos y mejorarlos, antes de iniciar la programación.

[94] Esta es una reflexión clásica de la ingeniería del software: ¿Tenemos que involucrarnos como ingenieros en sistemas a optimizar también los procesos no tecnológicos, si normalmente nos contratan para desarrollar un software?

semana. Si restamos todas las otras actividades que hace un analista diariamente, el tiempo real disponible para este análisis es corto y difícilmente nos da la oportunidad de convertirnos en 'expertos' en el tema; y si así lo quisiéramos, lo más probable es que no justificaría realizarlo, porque el proyecto ni siquiera ha sido aprobado y en la mayoría de los casos sería difícil obtener un pago por realizar este tipo de estudio preliminar. Es por ello que ciertas entidades de gobierno para evitar este problema muchas veces contratan proyectos de consultoría, específicamente para realizar el levantamiento de requerimientos del software[95].

En resumen, es realmente muy difícil determinar un alcance exacto para colocar en la cotización de un proyecto de software, en donde sabemos de antemano que no tenemos la información completa, no tenemos el tiempo suficiente para aprenderlo, y además tenemos la certeza de que variará su alcance en el transcurso de su ejecución. Por el otro lado el cliente exige una definición exacta de tiempo y presupuesto como requisito para auspiciar el proyecto.

Este escenario es poco alentador y en mi opinión es el que lleva a los responsables de esta etapa a cometer tantos errores que se traducen en retrasos en la entrega al final del proyecto.

La primera recomendación es que en el poco tiempo disponible para la elaboración de la cotización, el analista debe profundizar lo más que se pueda en el conocimiento de la "lógica del negocio", sin circunscribirse solamente en

[95] Lo malo de esta práctica en cambio, es que muchas veces se les prohíbe que sean ellos mismo los que desarrollen posteriormente el sistema. Lo que sucede después es que viene un nuevo equipo de profesionales que inicia la programación sin los conocimientos del análisis, y enfrentan el proceso solamente con la especificación de un documento frío.

lo presentado por el cliente. Sabemos que con esto no se convertirá en experto, pero será suficiente para iniciar una primera ronda de preguntas aclaratorias al mismo cliente, y verificar si sus requerimientos y reflexiones están completamente sustentadas.

Para el caso de los zapatos deportivos, probablemente encuentres que los conocimientos comúnmente aceptados en un módulo contable o de impuestos disponible en el mercado no son suficientes para un escenario de una zapatería especializada, y sea necesario programar una que otra herramienta adicional para controlar ciertos tipos de casos no regulares.

Hay que tener en cuenta que estos casos especiales se conocen solamente luego de conocer intensamente el giro del negocio. Por lo tanto es importante investigar mucho y evitar que se asuma que tiene los mismos conocimientos del cliente.

En resumen: **“APRENDER DEL NEGOCIO”**

LECCIÓN 2

Cuando un analista se inicia en el manejo de las reuniones de levantamiento de requerimientos, muchas veces no logra identificar las trampas lingüísticas[96] utilizadas por los clientes durante sus explicaciones, que pre-condicionan consciente o inconscientemente al analista en su medición de las variables más críticas del proyecto, que son el alcance, el tiempo y el presupuesto.

[96] Se denomina como “trampas lingüísticas” a aquellas frases que el cliente usa, la mayoría de veces sin una mala intención, para establecer ideas tergiversadas sobre un requerimiento de software, alterando la percepción del analista sobre la magnitud de un requerimiento. Esta tergiversación muchas veces se origina en el desconocimiento del cliente sobre lo que está pidiendo.

En el caso presentado, se colocaron las trampas lingüísticas más comunes utilizadas por los clientes:

- Al respecto del alcance, el cliente dijo: “esta herramienta no es nada fuera de lo común”. Esta conjetura nos crea una competencia irreal, y pre-condiciona la aceptación de un requerimiento sin profundizar su análisis. Esto coloca en una posición de riesgo crucial al respecto de un potencial atraso de un proyecto porque nos pone en condición de aceptar requerimientos complejos. Se ha visto que cuando se hace el levantamiento de requerimientos, el cliente considera “fáciles” hasta las cosas más complejas.
- De la misma manera al respecto del alcance, el cliente dijo: “Esto lo puedo hacer hasta en Excel”. Esta conjetura nos obliga a aceptar requerimientos basados en el hecho de que herramientas de la competencia son básicas y simples, cuando a veces no lo son. Ciertas veces el integrar un sistema informático con una herramienta común como Excel puede tomar mucho trabajo, no se diga de otras herramientas.
- Al respecto del tiempo, el cliente dijo que deseaba “iniciar antes de Junio”. Esta conjetura nos crea un escenario pre-condicionado de limitación de tiempo. El problema de estas aseveraciones no es el hecho de que no se pueda hacerlo, es el hecho de que el planificador trata de “acomodarse” a lo pedido por el cliente sin hacer un estudio serio del cronograma. Hay que tener en cuenta que si un proyecto requiere su ejecución en menor tiempo, deberá incrementarse su presupuesto o reducirse su alcance.

- Al respecto del presupuesto, el cliente dijo “unos jóvenes acaban súper rápido”. Esta conjetura lleva a pensar que se cuenta con un entorno de competencia con profesionales que al no tener gastos administrativos generales[97] ni el nivel de experiencia, pueden proponer un precio mucho más bajo que el de una empresa, y por ende pre-condiciona a colocar márgenes de utilidad menores. Se ha encontrado que esta conjetura varias veces es mencionada por los clientes de forma ficticia, para forzar al analista a ajustar su presupuesto.

Si un planificador de software se deja llevar de estas trampas en la etapa de definiciones iniciales, tiene una alta probabilidad de tener problemas de desfase de tiempo y presupuesto durante la ejecución.

Es por ello que se recomienda que se evalúe el lenguaje utilizado en las reuniones de levantamiento de requerimientos, se detecte todas las trampas lingüísticas y deliberadamente se retiren del proceso de análisis.

Hay que tener en cuenta que muchas de estas trampas pueden formar parte de los requerimientos fundamentales del proyecto, como por ejemplo las que condicionan el tiempo disponible de desarrollo del software, por ejemplo cuando el cliente necesita el software funcionando en una fecha específica para aprovechar una oportunidad de mercado. Igualmente no debe influir en la medición, pero probablemente se necesite un ajuste en el momento de aplicar el modelo matemático propuesto más adelante en este capítulo.

En Resumen: “**AISLAR LOS PRE-CONCEPTOS**”

[97] Se refiere a competencia establecida como Personas Naturales o grupos de profesionales en el trabajo al estilo “Freelance”, que en general tienen bajos costos administrativos y de operación.

LECCIÓN 3:

Uno de los problemas más complicados de la etapa de cotización, es el hecho de establecer la redacción del alcance del proyecto. Es muy dificultoso que en esta etapa se pueda elaborar un documento que explique detalladamente cada uno de los aspectos que incluirá el software, y si se lo intentara hacer, probablemente tomaría un esfuerzo similar a hacer el software.

Es por ello que en la redacción del alcance siempre queda un nivel de incertidumbre y apertura a las variaciones. Para muchos, esto significa un riesgo muy alto, y de hecho lo es, porque se deja la puerta abierta para que el cliente pueda "asumir" temas no considerados inicialmente, y lo peor de todo es que habría un justificativo formal para incluirlos.

Lastimosamente para cuando se firma un contrato de software, lo único que queda formalizado es el texto del alcance que fue propuesto por el mismo oferente. Entonces no se está protegido ante las modificaciones que ocasionan retrasos y fluctuación del tiempo. Imaginemos que dentro de nuestra propuesta colocamos el siguiente requerimiento:

- "Pantalla de Administración de Usuarios"
    - "Esta pantalla permitirá crear, modificar y eliminar los datos principales del usuario, manejar sus contraseñas de acceso y cargar su fotografía."

Para muchos, este texto explicaría bastante de lo que va a hacer el software en esta pantalla, sin embargo, es claro que esta descripción no dice exactamente cómo va a ser la pantalla que manejarán los usuarios, qué campos tendrá, qué

validaciones y consecuencias se tendrá de su programación. Esto significa que en un momento específico del proyecto, el cliente tranquilamente puede decir que falta colocar el campo de “correo electrónico” y que el sistema debería enviar un correo electrónico cuando la ficha del usuario es creada. Si volvemos a leer el texto arriba, no podríamos decir que “no está incluido” dentro del alcance, porque podría decirse que es parte de los requerimientos de creación de un usuario.

Por el otro lado, si se quisiera evitar una posible variación futura y describir exactamente cómo funcionará la pantalla, sus campos e interacciones, el estudio se extendería por mucho tiempo, y sería necesario desarrollar el diagrama entidad – relación, el diseño de las pantallas y la secuencia a seguir para cada uno de los botones.

La reflexión sería que si se va a hacer ese esfuerzo tan detallado para documentar un sistema, tal vez es mejor realizar de una vez con código fuente esta documentación, y tener ya una aplicación prototipo, pero, implicaría que debemos desarrollar una parte del sistema para poder cotizar, lo cual es totalmente insostenible.

También hay que tener en cuenta que los planteamientos realizados en el alcance de la cotización será uno de los aspectos más importantes para que nuestra propuesta sea la ganadora a ojos del cliente, y uno de los puntos seguramente evaluados será la “flexibilidad” de la propuesta. Se ha visto que los clientes rechazan las propuestas rígidas, y valoran mucho a los proponentes que tienen capacidad para adaptarse a los cambios.

Las recomendaciones para redactar el alcance dentro de la propuesta son las siguientes:

- Dividir el alcance del proyecto en módulos, y cada módulo en secciones y cada sección en pantallas o procesos.
- Para cada pantalla o proceso, explicar de forma descriptiva la misión de dicha funcionalidad.
- Al final del texto, colocar un párrafo que indique que todos los requerimientos plasmados en el alcance, serán formalizados en el documento de especificaciones funcionales, y que la verificación de la entrega final sea en base a este documento, reemplazando lo señalado en la cotización y el contrato.

El último punto es clave porque a pesar que no permite proteger en el detalle del texto del alcance, sí permite aplazar el tiempo de establecer formalmente los requerimientos definitivos. Lo importante es asegurarse que esta condición esté presente en la cotización y el contrato del proyecto, de tal manera que se tenga un sustento legal para modificar el alcance durante la ejecución, pero a propia discreción.

Esta recomendación podría parecer contraproducente en el sentido de que estamos asumiendo que las puertas quedarán abiertas y podrían solicitarse nuevos requerimientos que resulten en retrasos. Es verdad que esto no nos protegerá de los nuevos requerimientos, porque se ha demostrado que las variaciones son intrínsecas de los proyectos de software y creer que no existen sería ingenuo. Más bien la ventaja de esta recomendación es que nos permite manipular este riesgo posteriormente, facilitando la tarea de elaborar una

cotización ganadora sin una inversión grande e iniciar el proyecto, y establecer el documento de especificaciones funcionales en el tiempo que nos parezca más adecuado durante la ejecución del proyecto.

Por último, cabe destacar que en la encuesta realizada se encontró que un alto porcentaje de las empresas de software ecuatorianas, evitan ser muy rígidas e inflexibles con los clientes, adaptándose a los cambios propuestos durante la ejecución, pero con sustentos suficientes para la negociación de tiempos, si así se lo requiriese, entregando así un mejor servicio.

En resumen: "**APLACE EL RIESGO**"

LECCIÓN 4:

Uno de los puntos críticos dentro de una cotización de software es el hecho de colocar el tiempo exacto en el documento a entregarse, especialmente teniendo en cuenta que lo que se establezca en este momento puede significar un potencial retraso, problemas con los presupuestos y hasta el fracaso del proyecto.

En este sentido, se han desarrollado múltiples modelos matemáticos desde inicios de la era computacional, que buscaban encontrar una medición aproximada o exacta del tiempo que debía colocarse, basándose en múltiples variables intrínsecas del equipo de trabajo como por ejemplo el nivel de productividad de cada programador, medido en el número de líneas de código escritas[98] o a través de estadísticas de trabajos anteriores[99].

[98] Considero que la utilización de la métrica por número de líneas de código es muy negativo, porque muchas veces ocasiona que los desarrolladores en lugar de optimizar los algoritmos para que sean más pequeños y eficientes, escriben una gran cantidad de líneas de código para "mejorar su métrica".

El método planteado en este estudio es una combinación entre los modelos matemáticos, la experiencia y la opinión interna del equipo de trabajo. Para poder colocar el tiempo al proyecto, es necesario cumplir con los siguientes pasos:

- Divida el proyecto en módulos, los módulos en secciones, las secciones en pantallas o procesos. En este punto siempre hay dificultad para encontrar un listado completo. Lo que hay que asegurar es que se trate de un listado lo más amplio posible, y las variaciones de requerimientos serán tratadas aparte. En este listado no se deben incluir las tareas adicionales de análisis, diseño, documentación, pruebas, capacitación o instalación, solamente las tareas específicas de desarrollo.
- Con la descripción de cada una de las pantallas del sistema, ubique en Excel cada uno de los requerimientos, y a su lado cree una columna llamada "tiempo en horas". Tome al desarrollador más experto y al desarrollador menos experto del equipo, y pídales que en diferentes hojas para cada uno de ellos llenen la columna del tiempo, indicándoles que se trata del tiempo que necesitaría para cumplir cada tarea si es que tuvieran que hacerla ellos sólos. Llene usted también la hoja independientemente y al final promedie los tres valores para cada requerimiento[100].
- Ingrese estos valores en el archivo de Excel del ANEXO 4. El archivo de Excel lo que hará es sumar el número total de horas hombre requeridas

[99] Utilizar estadísticas no es malo, sino que se necesitan la recopilación de métricas al menos unos 5 proyectos anteriores. Para una empresa nueva o que inicia con la metodología, tendrá la oportunidad de fallar 5 veces, previo a tener un método de aproximación más cercano. Para muchas empresas esto puede significar unos dos o tres años.

[100] Otra forma de hacer esta misma idea es a través de la técnica llamada Planning Poker, descrita en el marco teórico de esta investigación.

para cada tarea, y aplicará unas estadísticas estándares para el cálculo de las otras etapas del desarrollo de software[101], y para controlar el riesgo sobre las variaciones, resultando en un tiempo preliminar, como si se fuera a trabajar con una sola persona.

- A continuación, en la parte inferior del mismo archivo de Excel, se encontrará un cuadro de porcentualización del tiempo, en función del número de personas asignadas al mismo. Esta tabla estadística[102] permite evitar la típica falacia que dice que "si coloca un nuevo programador, el tiempo se reduce a la mitad, si coloca otro se reduce a un tercio, etc". En esta tabla existen porcentajes de reducción del tiempo en función del número de personas y llegan solamente hasta 5 programadores. El motivo de esto es porque en primer lugar, en la teoría administrativa general se señala que en equipos de más de cinco personas, la comunicación y el choque cultural / social entre los integrantes ocasionarían obligatoriamente problemas internos. Si se trata de proyectos que requieren más personas, será necesario dividir la programación en varios equipos separados, de la misma manera con un máximo de cinco personas por grupo.
- A continuación se presenta en análisis de sensibilidad para los diversos escenarios de dos a cinco personas, creando una ficha de datos de tipo administrativo, tales como gastos en honorarios, transporte,

[101] Las estadísticas colocadas, son parte de mi experiencia en los proyectos que he desarrollado en el Ecuador, y servirán de punto de base para que luego las empresas puedan ir ajustándolos a sus propias métricas.
[102] Esta tabla está basada en mi experiencia en los proyectos que he desarrollado en el Ecuador y servirán de punto de base para que luego las empresas puedan ir ajustándolos en base a sus propias métricas.

alimentación, suministros, gastos de mantenimiento, de oficina y otros rubros varios.

- Una vez que se tienen los resultados para cada uno de los escenarios, se hace una tabla de resumen y un gráfico comparativo. En este momento será necesario tomar en cuenta los requerimientos del usuario al respecto de las limitaciones del tiempo. En el caso redactado, el cliente quería que el proyecto se termine hasta “Junio”. Basado en esta tabla será posible indicarle que si quiere hasta tal fecha, serán necesarios más programadores y su costo será mayor, o será necesario reducir el alcance. Como puede verse, este archivo de Excel también se convierte en un método para definir el presupuesto, y que con los cuidados del caso, podría utilizarse durante las sesiones de negociación para ir variando y proponiéndole opciones al cliente.
- Es muy importante establecer también dentro de la propuesta, la forma de pago requerida durante el proyecto. Muchos de los proponentes del software se preocupan mucho de los aspectos técnicos de la propuesta, pero rara vez se ha escuchado que se realice una proyección a futuro de los pagos del proyecto a través de un flujo de caja. Resulta ser que si no se hace un análisis profundo al respecto de las formas de pago, pueden llegar momentos específicos del proyecto en donde la empresa no tiene capital para trabajar, y tiene que obtenerlo de otras fuentes para poder continuar. Fuera de lo que se podría pensar, esto puede generar grandes desfases y retrasos en los proyectos. La propuesta es que se realice un flujo de caja del proyecto considerando las formas de pago y se busquen los puntos negativos a lo largo del proyecto. Si se encuentra

alguno, será necesario pedir una mayor cantidad de anticipo, o por el contrario, asumir ese gasto financiero dentro del flujo de operaciones de la empresa.

Una vez completa esta metodología, se tendrán definidas las variables de tiempo, presupuesto y forma de pago para cada uno de los proyectos de software.

Este método no debe ser considerado como estático, sino por el contrario, el planificador tiene la responsabilidad de ajustar constantemente las estadísticas y métodos, en función de su propio escenario de aplicación y la experiencia obtenida en su ejecución de proyectos.

En Resumen: "**PROFUNDICE EL ESTUDIO DE TIEMPO, PRESUPUESTO Y FORMA DE PAGO**"

LECCIÓN 5

Es posible que uno crea que la meta de la cotización está cumplida cuando uno ya tenga definidos los tiempos, presupuestos y formas de pago, sin embargo, se ha visto en el medio múltiples cotizaciones que no valoran económicamente como un adicional o condicionan ciertas cosas como lo siguiente:

- Método y forma de licenciamiento, limitaciones de uso, de instalación, de cesión, de uso de versiones.
- Se entrega los códigos fuentes, con o sin derecho de modificación, con o sin derecho de publicación posterior

- Derechos de propiedad intelectual, propiedad moral o propiedad comercial[103].
- Protección a los profesionales propios
- Garantía y soporte sobre el producto desarrollado
- Políticas de privacidad
- Requisitos para las capacitaciones
- Licenciamiento de terceros

Cuando no se condicionan estos aspectos, se da la apertura a que el cliente pueda asumir muchas de estas cosas como "incluidas" dentro de la propuesta, y por ende, ocasionarnos múltiples problemas posteriores, principalmente en el ámbito del presupuesto del proyecto y de la oportunidad de negocio sobre el software.

Para esto se ha preparado en el ANEXO 3 una plantilla que incluye algunas buenas prácticas a utilizarse como base para la presentación de una cotización de software.

Especialmente el código fuente, es uno de los temas más importantes dentro de la negociación, porque proporcionará la ventaja a la empresa de reutilizar dicho código fuente para nuevos métodos, investigaciones y productos. Otorgar al cliente todo el derecho de uso, publicación y comercialización de dicho código fuente, restringe la oportunidad de realizar ganancias adicionales para la empresa, constituyendo una especie de "lucro cesante". Actualmente se desconoce algún método exacto que permita valorar este tipo de derechos de

[103] Se refiere a la propiedad intelectual al propietario de la creación. Se dice propiedad moral al autor específico de la obra, que no necesariamente tiene derechos de propiedad intelectual. Se dice propiedad comercial, a aquel que ha adquirido derechos de comercialización de la obra, sin necesariamente ser su propietario o autor.

cesión del código fuente, sin embargo, cuando el cliente ha propuesto, se ha incorporado un valor entre el 20% y 40% adicional al valor del contrato, aunque dependiendo del escenario, podría llegar a ser mucho más, si es que la empresa además solicita exclusividad.

En Resumen: "**INCLUYA LIMITACIONES**"

LECCIÓN 6:

> "A comienzos de los años 80, el Reino Unido, Alemania, Italia y España anunciaron que trabajarían en conjunto para fabricar el Eurofighter, un avanzado avión de combate. El costo estimado del proyecto era de US$ 20.000 millones y se programó la entrada en servicio del avión para el año 1997. Hoy[104], después de más de dos décadas de problemas técnicos y gastos inesperados, la aeronave aún no está lista y su costo proyectado se ha más que duplicado, alcanzando una cifra cercana a US$45.000 millones."
>
> "Según la teoría económica estándar, el alto índice de fracasos se puede explicar fácilmente: la frecuencia de tan pobres logros es un resultado inevitable de los riesgos racionales que toman las empresas en situaciones de incertidumbre."
>
> "Después de estudiar este fenómeno desde dos perspectivas muy diferentes –la de un académico del área de los negocios y la de un psicólogo- hemos llegado a una conclusión distinta: no creemos que el alto número de negocios fracasados sea el resultado de decisiones racionales que no resultaron bien. Al contrario, nos parece que es una consecuencia de los errores cometidos en el proceso de toma de decisiones. Al hacer pronósticos de los resultados de los proyectos, resulta muy fácil que los ejecutivos sucumban ante lo que los psicólogos llaman "la falacia de la planificación". Bajo su embrujo, los ejecutivos toman decisiones basadas en un optimismo delirante, en lugar de ponderar racionalmente los logros, las pérdidas y las probabilidades. Sobreestiman los beneficios y subestiman los costos. Imaginan escenarios de éxito y

[104] Este artículo fue publicado en el año 2003

hacen caso omiso de los potenciales errores y cálculos erróneos. Como consecuencia se dedican a iniciativas que tienen pocas probabilidades de cumplir sus plazos y presupuestos o, incluso, de producir jamás el retorno esperado."

"El exceso de optimismo de los ejecutivos tiene su origen tanto en sesgos cognitivos - los errores que la mente humana comete al procesar la información- como en las presiones organizacionales. Estos sesgos y presiones son omnipresentes, pero sus efectos pueden ser atenuados. Si los procesos tradicionales para elaborar pronósticos –que tienden a enfocarse en las capacidades, experiencias y expectativas de la empresa- son complementados con un análisis estadístico simple de iniciativas similares que ya se han llevado a cabo, los ejecutivos pueden conseguir una estimación mucho más precisa del resultado probable de un proyecto. Este tipo de visión externa, como la hemos denominado, proporciona un mejor control de la visión interna, que es más intuitiva y, de esta manera, se reduce la probabilidad de que una empresa corra ciegamente a invertir su tiempo y dinero en un desastre." (Lovallo & Kahneman, 2010)

Uno de los aspectos repetidos que se ha podido visualizar en los planificadores de software es el "exceso de optimismo" durante las etapas de cotización y planificación. Esto es evidenciado en el sentido de que se sobrevalora la capacidad productiva del equipo de desarrollo de software, las bondades de la tecnología y sus habilidades organizativas internas, creyendo que se tiene un equipo con muchas mayores destrezas y ventajas que el de otras empresas.

El problema de caer bajo este concepto, es que el planificador toma abierta y deliberadamente decisiones y compromisos con el cliente, sin tomar consciencia del nivel de esfuerzo requerido y de los riesgos de retraso o incumplimiento que representan.

Por ello, es muy importante mantenerse al tanto de las experiencias de proyectos de otras empresas del sector o cualquier otro punto de vista externo, de tal manera que sea posible aprovechar esa información para no caer en la "falacia de la planificación", y se planifique de forma acertada.

Se recomienda que la AESOFT[105] o Colegios Profesionales del área de tecnología en el país, levanten una base de datos con información general de los proyectos de software ejecutados, sus tiempos, personal utilizado, su concepto, su presupuesto y otros, que puedan servir a toda la industria ecuatoriana a evitar fracasos. Probablemente esta práctica incremente los niveles de competitividad de las empresas ecuatorianas.

En resumen: "**EVITE EL EXCESO DE OPTIMISMO**"

LECCIÓN 7:

"La aversión al riesgo es un concepto usado en el contexto de la administración, organización de la empresa, negocios y gestión. Se refiere a la preferencia de un inversionista que no desea someter sus inversiones financieras a altos riesgos, por lo que sus inversiones serán en instrumentos más seguros, considerando siempre la eliminación del riesgo aunque alcance una menor rentabilidad". (Eco-Finanzas)

Cuando un inversionista evalúa una posible inversión en cualquier negocio, una de las variables más importantes a analizarse es el riesgo inherente y sus probabilidades de ocurrencia.

[105] Asociación Ecuatoriana del Software

Cuando se trata de emprender un nuevo proyecto de software, hay que actuar de la misma manera que actúa un inversionista, evaluando los riesgos inherentes al proyecto y sus probabilidades de ocurrencia. Pero esta reflexión no solamente se refiere a los riesgos de la ejecución del proyecto y del software en sí, sino también los riesgos de trabajar en una determinada industria, con determinada empresa e inclusive con determinadas personas.

Siempre hay que tener en mente que un proyecto de desarrollo de software rara vez toma menos de 3 meses y se extiende hasta varios años, lo que significa que aunque no se quiera, se tiene que cultivar un nivel de interrelación interpersonal muy alto con el cliente, y durante su ejecución probablemente se verá experimentada la aplicación de muchos valores y principios.

Por ejemplo, si de antemano se conoce que la empresa que nos ha llamado para desarrollar software, previamente hizo prácticas “no leales” con otro de sus proveedores, aunque sea en otra área que no sea el software, se debería pensar dos veces el arrancar dicho proyecto. Los ingenieros en sistemas consideran a sus proyectos como sus “hijos técnicos”, y como padres no quisiéramos que caigan en malas manos. Hay que tener en cuenta que se dedicarán varios meses o años al proyecto, y si lo pensamos, la vida es demasiado corta para dedicarle tanto tiempo a un cliente que podríamos considerar que no vale la pena a futuro.

Es por ello que antes de culminar la cotización, una de las tareas más importantes es el realizar una averiguación profunda sobre el cliente, sus orígenes, sus principios y valores, su misión y visión, sus accionistas y

administradores, su capital social, la relación con sus clientes, sus servicios, sus proveedores, su competencia, y todas las referencias posibles[106].

En base a estas variables, decidir si se trata de un cliente que fomentará buena imagen a nuestra empresa y producto, y si se considera que tiene valores y prácticas similares a las nuestras. En caso que haya ciertas incompatibilidades, no quiere decir que se debe desechar inmediatamente el proyecto, sino que se debe actuar como los inversionistas cuando comprometen un alto riesgo: "incrementar la utilidad y arriesgarse". Evidentemente si el riesgo evaluado es muy alto, se deberá desechar amablemente el proyecto.

Es por ello que se puede destacar la frase escrita por J.K. Rowling que dice lo siguiente:

> "Es imposible vivir sin haber fracasado en algo, al menos que viva tan cautelosamente que es mejor no haber vivido". (Rowling, 2011)

En resumen: "**CONOZCA AL CLIENTE Y ARRIÉSGUESE**"

[106] Un método sencillo para obtener información de una empresa, es el realizar la investigación en las bases de datos públicas del SRI, de la Superintendencias de Compañías o Bancos, de las Cámaras de Comercio o Producción. Es impresionante la cantidad de información que se puede obtener cruzando estas bases de datos. En algunos casos es posible inclusive hacer una reconstrucción, para obtener el balance general o estado de pérdidas y ganancias.

Capítulo 7: Al respecto de la etapa de análisis y el documento de especificaciones

Escenario

Una vez que se entregó la cotización a Don Juan Zapatero luego de una semana desde la última reunión, se tomó prácticamente dos semanas para analizar y negociar el proyecto. Luego de contrastar nuestros ofrecimientos, con respecto a otras ofertas, nos notificó que nuestra oferta era la ganadora. En ese sentido se procedió a negociar las condiciones y firmar el contrato de provisión de software.

Don Zapatero ha pedido que se le explique la metodología a utilizarse nuevamente y que se arranque con la programación del sistema lo más pronto posible, pues quería ver resultados pronto, tal como todos los clientes piden al principio.

Le explicas entonces que se inicia la etapa de análisis, y por ende es necesario realizar diversos estudios sobre el negocio y el proceso a implementarse, para poder llegar al desarrollo de un prototipo funcional.

Sin entender mucho de los términos utilizados, Don Zapatero acepta tu planteamiento y pide que lo llamen cuando se tenga algo para revisar. Es hora de iniciar el trabajo, cómo lo hacemos?

Estudio de Buenas Prácticas

Una vez que el proyecto ha sido aprobado formalmente, el camino es más fácil, pues existen múltiples metodologías teóricas / prácticas utilizadas globalmente,

algunas muy bien documentadas por ciertas casas de software, que sirven para organizar las tareas, distribuir responsabilidades, generar documentación, realizar diagramas explicativos y otras tareas previas a realizar la programación.

Ejemplos de este tipo de metodologías son MSF, RUP, Extreme Programming, SCRUM, V-Model y otras[107], que normalmente presentan una secuencia de tareas predefinidas y un conjunto de plantillas de documentación obligatoria y opcional a realizarse para considerar el software "correctamente" documentado.

Lo importante en este punto es el "especializarse" en cualquiera de las metodologías, preferiblemente alguna que se enmarque dentro de las metodologías ágiles[108], revisar los pasos a seguir y la documentación que es obligatoria desarrollar. Inclusive busque en Internet si existen herramientas gratuitas o pagadas que sirvan para llevar adelante su metodología.

Una vez que lo tenga todo entendido, "personalícela" o "haga su propia versión". He visto que muchos analistas a veces sufren por ciertos requisitos obligatorios de las metodologías aunque parecieran no tan aplicables, mi punto de vista es que uno debe realizar la documentación justo a la medida de las necesidades, asegurándonos que cada palabra escrita agregue valor al proyecto.

Hay veces que para poder orientar adecuadamente el proyecto, se requiere un documento que establezca los acuerdos iniciales entre ambas partes, y un

[107] Si se desea mayor detalle al respecto de las metodologías disponibles, es importante revisar la documentación relacionada al marco teórico de la presente investigación.
[108] Esta recomendación nace del hecho de que las buenas prácticas presentadas utilizan varios de los conceptos de las metodologías ágiles.

documento de conceptualización general[109] previo al desarrollo del documento de especificaciones funcionales.

Cualquiera que sea el caso, y acercándonos más a las metodologías ágiles, se prefiere realizar la menor cantidad de documentación técnica posible, e invertir más tiempo en la programación. Esto es fundamentado en que si se tiene la certeza de que el software va a cambiar en el futuro, la documentación rápidamente va a quedar desactualizada, y el costo de regresar a actualizarla para cada cambio no tiene un justificativo suficiente, y menos aún cuando no genera valor agregado si es que su resultado ya está expresado en el propio software.

Seguramente esta aseveración hará que varios propongan un punto de vista contrapuesto, señalando que si no se tiene la documentación suficiente y el estudio completo, sería imposible iniciar con la programación. Inclusive muchos de los contratos de software solicitan que al final se entregue la documentación actualizada, aunque nunca es utilizada posteriormente. La experiencia ha dicho que con la idea inicial del proyecto y el diagrama de entidad – relación[110], para los desarrolladores ya es posible iniciar con la primera iteración de la programación, siempre y cuando ya haya sido definido el estudio arquitectónico y los estándares de programación.

Hay que tener en cuenta también que no es muy común encontrar un equipo de desarrollo de software que tenga las habilidades de redacción y documentación. Normalmente esta tarea es aburrida para un programador, la

[109] En la metodología MSF, este documento sería el de "Visionamiento"
[110] El diagrama entidad – relación debe ser realizado por los expertos del negocio y técnicos, revisado, generalizado y desnormalizado si es del caso. Mientras más detallado y pulido es el diagrama para la primera iteración, menos cambios en la programación se generarán posteriormente.

cual probablemente le tomará más tiempo del necesario, período que podría ser productivo en otras tareas. La otra opción es que la empresa contrate de forma separada un documentador del proceso, pero considerando sus costos adicionales.

Por último, y lo más importante, explíquele con lujo de detalles al cliente su metodología, de tal manera que tenga muy claras las actividades que se van a llevar a cabo, las prioridades en las iteraciones, los documentos que se desarrollarán, para que prepare su estructura organizacional y establezca una estratégica de preparación psicológica del personal. [111]

En resumen: "**APRENDA Y DIFUNDA SU METODOLOGÍA**"

LECCIÓN 9:

El contenido del documento de especificaciones funcionales rara vez cumple con un formato estándar, entre empresa y empresa, y proyecto y proyecto, y por ende se suele encontrar que no se incluye alguna información crítica que posteriormente es solamente tratada verbalmente.

Fuera de lo que se pensaría, la mayoría de estos puntos críticos no están relacionados directamente con el alcance del sistema, sino más bien con requerimientos no funcionales, tales como la infraestructura requerida, la arquitectura, el rendimiento, los mecanismos de prueba y los conceptos generales.

[111] Los procesos de desarrollo de software no solamente implican grandes esfuerzos del equipo técnico, arquitectos de software y demás especialistas, también significa que las personas que lo van a utilizar tienen que cambiar muchas veces sus esquemas de trabajo, una temporada de doble trabajo, digitación y migración inicial, validación de la información, etc. Es necesario que las personas involucradas estén preparadas para este proceso desde el principio, y así evitar tempranamente las normales resistencias a este tipo de proyectos.

Una vez la empresa tuvo un problema por este motivo, pues no había caído en cuenta que el documento de especificaciones funcionales no tenía descrita la infraestructura que el sistema necesitaba, y nunca se realizó un documento que planifique la capacidad tecnológica requerida. En el momento que se quisieron iniciar las pruebas, recién se dieron cuenta que el sistema tenía una arquitectura que requería un servidor adicional. Recién en ese momento buscaron presupuesto, corrieron procesos de aprobación, adquisición y configuración. Por este motivo, el proyecto retrasó su entrega. Si se hubiera documentado y socializado este requerimiento con anticipación, hubiera habido suficiente tiempo para cubrir estas tareas.

Para cubrir inconvenientes de esta naturaleza, se ha desarrollado un conjunto de plantillas para los documentos de acuerdos iniciales, visionamiento y especificaciones funcionales[112], que sin ser rígidos ni aplicables para todo tipo de escenario, servirán de guía y estándar para su elaboración.

En resumen: "**DOCUMENTE LO ESTRICTAMENTE NECESARIO**"

LECCIÓN 10:

Cuando se habló de la etapa de cotización, hubo un punto que indicaba que era mejor "aplazar el riesgo"[113] para la etapa de elaboración del documento de especificaciones funcionales. Podría decirse que ha llegado el momento en el que el cliente nos preguntará sobre la formalización de dicho documento, pues hay que tener en cuenta que estipulamos en el contrato que los requerimientos

[112] Estos documentos se encuentran adjuntos en el ANEXO 5
[113] Se refiere al contenido de la Lección 3.

de dicho documento serán los que se aplicarán formalmente en la revisión final del proyecto.

La idea es "seguir aplazando el riesgo". El planteamiento es que se ejecute lo más rápidamente posible la primera iteración del desarrollo del proyecto, llamándola "prototipo funcional". Se busca tener la primera versión lo más rápido posible. Cuando el cliente tiene algo que visualizar, recomendar y proponer cambios[114], aunque le falten algunas secciones funcionales, se olvidará temporalmente del documento de especificaciones funcionales formalizado, y preferirá ver el producto que se está elaborando[115], además hay que tener mucho en cuenta que para un usuario es más fácil visualizar un sistema tal como va a quedar, en lugar de un dibujo.

Para cada iteración del software se realizará una reunión y se le presentará al usuario los nuevos requerimientos implementados, el cuál en muchos casos, va formando parte activa en el pulimento de las pantallas, reduciendo radicalmente el tiempo requerido para las pruebas y validaciones de concepto.

Esta metodología también tiene sus desventajas, pues hay clientes que creen que con la primera o segunda iteración, ya se está al 90% del desarrollo del sistema, y proponen ya empezar fases de pruebas o fases de producción simuladas. Hay que procurar controlar este nivel de "emoción" del cliente, aceptando estas salidas solamente cuando se tenga confiabilidad de que el producto está completo.

[114] El gerente de proyecto debe tener muy claro los cambios que pueden asumirse y los cambios que deben ser desechados. Esto lo conocerá siempre y cuando esté profundamente involucrado con el modelo de programación y de base de datos. Es por esto que se dice que el gerente de proyectos debe ser un desarrollador con mucha experiencia, y no que tenga habilidades solamente en el área administrativa o de control de proyectos.

[115] Tuve un caso en donde el cliente me dijo "prefiero ver el software" que otro tipo de documentos. Luego revisaremos eso.

Una vez se cometió la equivocación de aceptar la salida a ambiente de producción de una sección del sistema que no estaba completa y probada, motivado en el hecho de satisfacer un pedido expreso del cliente. El resultado fue que las personas que lo usaron tuvieron una mala experiencia y los comentarios rápidamente se regaron a lo largo de toda la organización, y para la salida a producción real, las personas estaban ya llenas de pre-conceptos y de malos comentarios, que revertir esas ideas y reivindicar la confiabilidad del software terminó siendo una tarea muy ardua.

Ahora la pregunta sería, en qué momento completamos el documento de especificaciones funcionales?

La respuesta sería: en el momento en que nos encontremos en una iteración de la programación que ya nos presente un porcentaje razonable de la funcionalidad a ser implementada. Lo que debemos hacer es sencillo, debemos desarrollar el manual de usuario de la versión elegida, inclusive con capturas de pantallas, y eso colocar dentro de la sección del alcance del proyecto, así no correremos ningún riesgo contractual, además que dicho manual de usuario nos servirá probablemente para la entrega final.

En resumen: "**DESARROLLE UN PROTOTIPO E ITERE**[116]"

[116] Del término "Itere" tiene relación a uno de los conceptos fundamentales de la metodología SCRUM, que indica que es necesario hacer varios ciclos llamados "iteraciones", de tal manera que el software en cada uno de los ciclos presenta un producto final entregable.

Capítulo 8: Al respecto de la etapa de programación

Escenario

Don Luis Zapatero está un poco confundido, pues en la cotización y las reuniones posteriores se le presentó una metodología secuencial de tareas para el desarrollo de su sistema, pero en realidad, se han hecho muchas cosas paralelamente, porque sin darse cuenta, parece que ya se inició la etapa de programación, sin haberse terminado la etapa de especificaciones funcionales.

Sin embargo, Luis está conforme con la metodología utilizada, porque él ya puede visualizar el software, y cada vez que hay reunión existen cosas nuevas y puede ver cómo funcionan y opinar sobre ellas. Él a sus adentros dice, “ellos deben saber lo que hacen, y lo importante es que ya estoy viendo resultados”.

¿Cómo debe manejarse al cliente durante la etapa de desarrollo?

¿Cómo debe manejarse al equipo de programación durante la etapa de desarrollo?

Estudio de Buenas Prácticas

LECCIÓN 11:

> “Tal vez la idea de que las empresas deban “encantar” a sus clientes se ha arraigado tanto porque “¡Superen las expectativas!” suena más atractivo que “Resuelvan el problema lo antes posible”. Pero si una empresa hace algo mal, yo espero una resolución rápida. Eso es todo. Si entrega extras, lo veo como una compensación por su error más que sentirme conmovido porque superaron mis expectativas.” (Ing, 2010)
>
> “En nuestra investigación, la pregunta era general, pero la conclusión fue clara: cuando le preguntamos a 75.000 clientes si son más leales a una empresa cuando trata de

> hacer lo imposible en las interacciones de servicio, respondieron “no realmente”. (Dixon, Freeman, & Toman, 2011)

En el momento de medir el alcance de un proyecto de software, los ingenieros en sistemas se dejan llevar fácilmente por las “mieles de la última tecnología”. Aprovechamos los nuevos proyectos para poder aplicar dicha tecnología, muchas veces sin un real sustento de hacerlo y su uso ocasiona también retrasos por la falta de conocimiento. Estas decisiones las escudamos bajo los motivos de proporcionar al cliente un excelente producto.

La pregunta es: ¿Es este esfuerzo valorado por el cliente?

La respuesta en la mayoría de los casos es “NO”. En el caso de desarrollo de software, al cliente le interesa que funcionen los requerimientos dentro del programa informático y punto[117]. No se sentirá “maravillado” si es que han utilizado la última tecnología, la mejor documentación, si han generalizado los componentes para la reutilización o si la arquitectura permite escalabilidad.

No con esto se quiere decir que debe dejarse de lado la estructuración de una buena arquitectura y tecnología para el proyecto. Tal es así que es muy recomendable que se utilice la mejor tecnología factible, si esto no va a representar ciclos muy extensos de capacitación y experimentación. Lo que se desea recalcar es que muchas empresas dedican mucho tiempo a realizar documentación, arquitectura, análisis y otros temas que no son directamente el software, que por más que se presentan como productos complementarios a los clientes, ellos no encuentran en estos un valor real en comparación con el software funcionando, que es a la larga el producto por el cual están pagando.

[117] Esto no es una regla general porque hay empresas muy cercanas a la tecnología, que sí se preocupan de estos aspectos, pero el concepto explicado en este punto sí aplica para la mayoría de los proyectos.

La recomendación es que se busque primero cumplir con lo que busca el cliente, para que pueda conocerlo, evaluarlo y mejorarlo tan pronto como sea posible, y luego se haga una nueva iteración que incluya los productos complementarios. Esto permitirá que el cliente siempre tenga algo que ver, facilitando así la capacidad de negociación en caso de que sea necesario extender el tiempo del proyecto.

Hay que tener en cuenta estas premisas cuando se planifica una iteración, pues la priorización de requerimientos mucho tiene que ver con la priorización del cliente. Por último, si se va a atrasar en una entrega, el cliente sabrá que lo que falta, son las funcionalidades laterales, y por ende tendrá una tendencia muy clara a continuar y negociar, en lugar de cerrar el proyecto.

En resumen: "**PRIMERO CUMPLA LA PROMESA BÁSICA**"

LECCIÓN 12:

Uno de los problemas a los que debe enfrentarse todos los días el gerente del proyecto durante la etapa de programación es al respecto de las variaciones de requerimientos, y éstas probablemente se vean intensificadas en la metodología de iteraciones propuesta anteriormente, porque se le da la oportunidad al cliente de opinar muy constantemente.

Durante la encuesta realizada en la presente investigación, se evidencia que existe un gran grupo de personas que francamente aceptaron su impotencia para evitar que estas variaciones ocurran, aunque también se reconoce que si no hay control sobre esto, existe una alta probabilidad de obtener un retraso y hasta de un fracaso.

Como se vio en la etapa de cotización, era en extremo complicado obtener un listado de requerimientos tan detallado como para que no quede la más mínima duda de lo que se incluirá en el software. Como no es posible protegerse por ese lado, se deja una puerta abierta para que el cliente siempre proponga mejores cosas, que seguramente toman tiempos adicionales.

Esto en el fondo no es malo, porque probablemente el mismo cliente encuentre que mientras se va progresando, se van planteando reflexiones que él mismo no se había planteado antes, o inclusive va encontrando mejores formas de hacer las cosas. Si vamos evolucionando sobre estos conceptos, se obtendrá al final un producto más práctico y cercano a las necesidades, y que para el cliente tenga el mayor retorno a la inversión[118].

El punto es que si no se limita esa "inmensa creatividad" del cliente, especialmente cuando hay usuarios que piden que se incluya un botón que diga "hacer mi trabajo de hoy", evidentemente que no se podrá terminar el proyecto dentro de los plazos estipulados en el contrato, y hasta es posible derrocharse vanamente las utilidades del proyecto.

Hacerle entender al cliente que sus variaciones tendrán impacto en el tiempo y presupuesto, es una tarea ardua y a veces imposible[119], y peor aún se complica cuando su actitud está enfocada a frases como: "yo sí le dije al principio del proyecto" o "esto se asumía que estaba incluido".

[118] Esto es sustentado en el hecho de que este proceso evitará la creación de nuevas versiones del sistema por cuestiones de corrección.
[119] En los contratos del sector público, es muy complicado que los auspiciantes asuman nuevos tiempos y presupuestos, dejando la responsabilidad total sobre el proveedor. Casos muy excepcionales son los que permiten la emisión de contratos complementarios.

Al no tener herramientas determinísticas y exactas que nos permitan delimitar con punto y coma los detalles a incluirse en el software, al parecer estamos atrapados en nuestra propia trampa metodológica.

Luego de analizar este problema se ha determinado que parte desde la misma percepción básica fundamental, pues es de aceptación general el ver al software como un "producto", cuando presenta algunas similitudes con los procesos de "consultoría" o "asesoría", es decir, que se parece también a un "servicio".

Cuando concebimos un servicio como si fuera un producto, surgen estas complicadas encrucijadas metodológicas, pues si queremos evitar las variaciones de requerimientos, estaríamos yéndonos en contra de una de las propiedades intrínsecas del software, que es el ser "evolutivo". Pero si abrimos la posibilidad de variar mucho, el proyecto probablemente fracase también por el exceso de complejidad y los retrasos.

A continuación se presentan algunas recomendaciones para controlar que las variaciones:

- Siempre que sea posible, colocar en el contrato del proyecto una cláusula donde se diga explícitamente que el cliente reconoce que el software es un proceso en constante evolución y por ende son posibles las variaciones durante el proyecto pueden generar variaciones en el tiempo y presupuesto del contrato, y que por ende se cuidará los márgenes de utilidad establecidos por el proveedor, si estos se vieren afectados por los cambios.

- Una de las clásicas recomendaciones dice que “todo debe estar firmado por el cliente”. En oportunidades en donde a pesar que estaba firmado por él, no le importaba mucho, y desconocía lo que se había acordado previamente. Hay que tener en cuenta que el cliente normalmente no es una persona técnica y firma los documentos bajo una confianza profesional, más que bajo un conocimiento exacto. Si en un momento se utiliza una firma suya en su contra, probablemente tenga un sentimiento similar a la “traición a su confianza”. Esto puede ocasionar muchos mayores problemas que la variación de un requerimiento específico. Hay que tener mucho tino cuando se utilizan las firmas como medio persuasivo y de negociación.
- Dentro del marco de la profunda responsabilidad profesional, si se le explica al cliente desde un principio que el software tiene posibilidad de variar, y existe la posibilidad de que el tiempo y presupuesto también varíen; él se preparará para este escenario, y pensará más profundamente antes de pedir algo. Para manejar esta técnica, es necesario contar con una amplia confianza del cliente.
- A pesar que parece un tema alejado del problema, es de vital importancia lograr que el cliente mantenga una actitud positiva hacia usted y una notable confianza técnica[120]. Esto le permitirá establecer una comunicación franca y facilitará una negociación de variación de cambios, controlando la flexibilidad y las restricciones. Si el cliente tiene una actitud negativa, puede ser una fuente de retrasos inminente, pues hasta en la revisión más sencilla se encontrarán trabas. El manejarse de

[120] Para esto también sirve el hecho de hacer ciclos iterativos cortos, que le presentan resultados rápidos a los clientes. Tenga en cuenta que los clientes valoran la ejecución en función de los resultados.

esta manera adicionalmente mejorará las posibilidades comerciales de la empresa, porque inclusive podría decirse que al cliente le "gusta" trabajar con usted y lo recomendará abiertamente.

- Basado en la experiencia personal, se ha desarrollado una técnica que me permite valorar el tamaño de un requerimiento, y las formas de solicitar una ampliación:
    - El primer requisito es que el gerente de proyecto esté profundamente involucrado con el proceso de desarrollo, tanto como para que conozca el modelo entidad – relación[121] y las pantallas del sistema.
    - Cuando el cliente solicita cualquier cambio, se le pide un tiempo para evaluar el impacto.
    - El gerente del proyecto se reúne con el programador más cercano al cambio, y evalúan juntos qué cambios en el modelo entidad-relación son los resultantes de dicho cambio:
        - Si en la evaluación se encuentra que el cambio no modifica el modelo actual, puede decirse que es un cambio "menor".
        - Si modifica solamente la estructura de campos puede decirse que es un cambio "medio".
        - Si involucra la creación de nuevas tablas o relaciones, puede decirse que es un cambio "alto".
    - En la reunión con el programador también se analiza las pantallas y procesos.

[121] En el modelo entidad – relación normalmente se plasma la "complejidad" del sistema. Es por ello que se pide que en la primera iteración se elabore este diagrama con la mayor cantidad de detalle posible.

- Si el cambio involucra la modificación de una sola pantalla, puede decirse que es un cambio “menor”.
- Si el cambio involucra la modificación de hasta 3 pantallas, puede decirse que es un cambio “medio”.
- Si el cambio involucra la modificación de más de 3 pantallas o la creación de nuevas pantallas, puede decirse que es un cambio “alto”.

- Se promedia los valores de la evaluación del modelo entidad – relación y las pantallas, y se toma la decisión de la magnitud del cambio. Es posible también que se coloque una columna adicional con el número de horas requeridas para el cambio.
- Una vez que se ha evaluado un cambio, es muy importante crear un documento de control de cambios, de tal manera que el cliente lo lea con detalle y firme su aceptación, si es que realmente desea que se incorpore.
- Se lleva una hoja de control sobre las hojas de control de cambios firmadas, dejando la flexibilidad de aceptar 2 cambios altos, 3 medios y 5 menores[122]. Si se supera esto, es momento de formalizar un pedido de modificación del plazo del proyecto con el suficiente sustento. Se utiliza esta metodología para evitar la sensación de rigidez ante el cliente, y estar cada rato en trámites desgastantes de modificación del alcance.

[122] Cada empresa puede determinar sus niveles de flexibilización, muchas veces en función del tamaño del proyecto. Se asume que los cambios de esta naturaleza no deberían superar el 10% del tiempo total de la programación.

- Esta metodología la debe conocer y aceptar el cliente, en el documento de acuerdos iniciales, y hacerla respetar con amabilidad y firmeza.

En resumen: "**VIGILE LAS VARIACIONES**"

LECCIÓN 13:

Hay algunas variables que pueden influir en el retraso, que están motivadas en los métodos de programación y arquitectura utilizadas en el proyecto. Es importante que se haga una profunda reflexión técnica sobre estas variables antes de empezar la programación de la primera iteración.

En este sentido, es recomendable contratar un servicio de desarrollo de arquitectura, si es que no se tiene un arquitecto de software interno, para que elabore las estructuras e interrelaciones técnicas del sistema, y cumpla el rol de fuente de consulta técnica y resolución de conflictos tecnológicos, e inclusive hasta la etapa de salida a producción.

En el Internet se han documentado una gran cantidad de errores comunes de programación y arquitectura, que son los denominados "anti-patrones" del software. El arquitecto de software o líder técnico del proyecto tiene la responsabilidad de revisar y conocer cada uno de los anti-patrones, e identificar las estrategias de mitigación, desde el principio del proyecto.

El arquitecto de software también tiene la responsabilidad de preocuparse por la "productividad" del equipo de desarrollo. Es por ello que se le requerirá como un producto indispensable de su trabajo, la elaboración de plantillas de generación de código y otro tipo de herramienta adicional que permita que el

programador no pierda el tiempo en la tecnología y se concentre en dar solución al problema del negocio del cliente.

En resumen: "**TECNIFIQUE LA PROGRAMACIÓN**"

Capítulo 9: Al respecto de la gestión del personal

Escenario

La ejecución del proyecto del software para la fábrica de zapatos deportivos de Don Zapatero iba muy bien, y cumpliendo cada vez más las expectativas de los usuarios que lo probaban.

Sin embargo, el equipo de trabajo está presentando algunas dificultades internas, que aunque no son todavía públicas y no las conoce el cliente, preocupan bastante al gerente del proyecto.

Las principales preocupaciones están en que ingresó al mercado una empresa multinacional que tiene la capacidad de pagar más alto que el promedio a los profesionales, y está tentando directamente a tu gente a cambiarse de trabajo, y ya corre un rumor de que podrían irse dos personas antes de culminar el proyecto de don Zapatero.

Se suma adicionalmente un problema que todavía no ha explotado, pero que preocupa también al gerente de proyecto, y es que ha notado que su gente no trabaja con el mismo ánimo que cuando ingresó a la empresa hace cinco años. De las averiguaciones directas, se pudo averiguar que tal vez una de las causas es una vieja pelea entre dos integrantes del equipo que está empezando a trascender del ámbito personal al profesional, y está afectando a todos.

¿Qué hacer para mejorar las interrelaciones del equipo interno y su motivación general?

¿Cómo fomentar un clima laboral adecuado para la productividad?

Estudio de Buenas Prácticas

LECCIÓN 14:

> "Frente a las intensas presiones del mercado, a menudo las corporaciones abarcan más de lo que pueden manejar: aumentan el número y la velocidad de sus actividades, elevan las metas de desempeño, acortan los ciclos de innovación e introducen tecnologías de gestión o nuevos sistemas organizacionales. Durante un tiempo tienen muchísimo éxito, pero con demasiada frecuencia el CEO[123] trata que este ritmo vertiginoso se convierta en la norma. Lo que empezó como una explosión excepcional de logros se vuelven una sobrecarga crónica, con consecuencias nefastas. El ritmo frenético no sólo socaba la motivación de los empleados, sino que el foco de la empresa se dispersa en varias direcciones, lo que puede confundir a los clientes y amenazar la marca."
>
> "Al darse cuenta de que pasa algo, frecuentemente los líderes intentan combatir los síntomas en lugar de la causa. Y debido a que interpretan la falta de motivación de los empleados como pereza o una inclinación a quejarse, por ejemplo, aumentan la presión, lo que empeora las cosas. El agotamiento y la resignación comienzan a expandirse por la empresa y los mejores empleados se marchan".
>
> "Nosotros llamamos a este fenómeno la *trampa de la aceleración.* Daña a la firma en muchos niveles. Según nuestra investigación, a las empresas sobreaceleradas les va peor que a sus pares en desempeño, eficiencia, productividad y retención de los

[123] Se refiere a las siglas de Chief Executive Officer. Se refiere a la máxima autoridad administrativa de una empresa, correspondiendo al Presidente o Gerente General. El término es muy utilizado en Estados Unidos y otros países de habla inglesa, aunque ha trascendido también al español.

> empleados, entre otros indicadores. Es un problema muy común, especialmente en el actual entorno de disponibilidad total y reducción de costos.” (Bruch & Menges, 2010)

Uno de los motivos de desmotivación más comunes dentro de un equipo de trabajo es efectivamente la “sobre-asignación” de tareas. La idea es que si el gerente de proyecto exige a los empleados que constantemente inviertan el mismo esfuerzo acelerado todos los días, la energía y motivación decaerán y el desempeño profesional también caerá.

Esto sucede cuando el gerente de proyecto siente que el proyecto se está “atrasando” o se ha aceptado incorporar nuevos requerimientos al proyecto para ser ejecutados dentro del mismo plazo, presiona a los empleados a incrementar el ritmo de trabajo, obteniendo mejoras momentáneas en los resultados, pero si no se detiene ese ritmo acelerado, el equipo se cansa y genera problemas más difíciles de manejar como es el caso de la desmotivación y el cansancio crónico.

En el medio ecuatoriano es muy común encontrar empresas que llegan a exigir una ampliación de horario de trabajo de los programadores, a veces hasta el amanecer, muchas veces bajo la simple excusa de que “así es este trabajo” o como acto de “demostración del compromiso con la empresa”.

Es importante reconocer y reflexionar que la productividad de un desarrollador de software, no necesariamente está ligado al número de horas que se encuentra frente al computador, sino al respecto del número de momentos de lucidez que logra aprovechar durante el día. Una persona cansada disminuirá

notablemente estos momentos de lucidez, y por ende, generará menores resultados que su promedio normal.

Por el otro lado, si es que no existe un nivel de presión adecuado, los empleados no asumirán el reto profesional, divagarán y se distraerán fácilmente con otros temas, entendiendo que las condiciones del proyecto son muy holgadas.

En la redacción del caso, se plantea también otra fuente de desmotivación, que viene más bien ligada a un ambiente laboral no muy adecuado. Esto es dado cuando el equipo de trabajo no genera un ambiente afable y jovial que facilite las comunicaciones y la colaboración, que son tan necesarias durante este tipo de procesos.

Existen también fuentes de desmotivación originadas desde la política empresarial, como son el sentimiento de que la remuneración económica no es compensatoria, que exista el sentimiento de estancamiento profesional, que la distribución del trabajo entre pares esté mal equilibrada, o que el esquema de justicia interna no resulte ser “tan justo”, entre varias otras.

Lo más importante para resolver los problemas de motivación es el conocer sus orígenes y mitigarlos de forma directa, abierta y participativa.

En la encuesta desarrollada en la presente investigación se encontró un dato que es curioso y que puede ser aplicado como una práctica útil. La encuesta demostró que los equipos de desarrollo de software se motivan más con un plan de capacitación y promoción profesional frecuente, antes que con una

motivación de tipo económica. Esta práctica motiva al personal y la empresa gana en productividad general, gracias al uso de mejores tecnologías.

Lo importante es que la gerencia del proyecto, tiene mucho de su responsabilidad en valorar la "salud emocional" de los integrantes del equipo y anticiparse a los problemas, pues a la larga son ellos quienes ejecutarán el proyecto y lo llevarán al éxito o fracaso. Omitir esto puede generar un descontrol general y ocasionar inclusive dramáticos retrasos.

En resumen: "**MOTIVE ESTRATÉGICAMENTE**"

LECCIÓN 15:

> "El Mundial de Fútbol Sudáfrica 2010, al representar una competencia entre "empresas" similares donde la gestión del talento es un factor crucial, puede ser tomado como analogía para analizar la competencia entre empresas diferenciadas sólo por las capacidades de sus miembros. El Director Técnico define un sistema de juego en función de los jugadores con los que puede contar. Después selecciona entre todos los jugadores que pueden representar su país a aquellos que mejor se ajustan al esquema de juego que pretende. Si bien los clubes son los principales inversionistas en el desarrollo de jugadores, las asociaciones nacionales también aportan lo suyo. Organizan entrenamientos especiales para los juveniles, impulsan campeonatos como el Sub-20 o el Sub-17 y organizan partidos amistosos con el fin de probar jugadores y de consolidar su equipo nacional."
>
> "La competencia entre empresas estructuralmente similares en el Mundial de Sudáfrica 2010, nos ofrece valiosas lecciones para mejorar la gestión del talento en el contexto empresarial. Si bien los equipos con una base de talento mayor obtuvieron mejores resultados, también hubo varias selecciones nacionales que lograron resultados que no se explican a partir de la calidad de sus jugadores individuales. En otras palabras, el desempeño organizacional no se explica a partir del talento individual de los

> empleados. Depender de mega-estrellas individuales tampoco favorece el desempeño. Para desgracia de sus equipos, varios jugadores estelares se deslucieron en el Mundial. Por el contrario, los mejores jugadores del torneo tienen en común que pertenecían a equipos que funcionaron como conjunto. Finalmente, si bien es importante contar con ejecutivos talentosos, el caso del entrenador de la selección inglesa –Favio Capello- sugiere que el talento ejecutivo agrega poco cuando no está sintonizado con la organización en que es empleado." (Hermans, Sioli, & Fay, 2010)

Al igual que funciona en un equipo de fútbol, en donde se tiene un arquero (arquitecto), defensas (base de datos), medios (comunicaciones) y delanteros (interfaz de usuario), en el equipo de software se tiene profesionales que están en el mismo juego pero que cumplen roles diferentes y complementarios.

La reflexión dejada por el artículo propuesto por Hermans es que de la misma manera que en un equipo de fútbol, en donde si un medio no puede hacer un pase a un delantero, nunca se llegarán a hacer los goles; en el ámbito del software funciona de maneras muy similares, en donde "el pase" significaría la facilidad de comunicación e interacción entre los participantes para lograr un objetivo común.

Entonces es responsabilidad del gerente de proyecto como director técnico, el estructurar un equipo de profesionales con alto potencial sinérgico para ejecutar el proyecto desde inicio a fin, como si fuera una temporada del campeonato, aunque durante este transcurso muchas cosas pueden pasar: por ejemplo la lesión de uno de los jugadores, el cambio del planteamiento de ataque o hasta el cambio del técnico.

Traer una “estrella” al equipo puede ser contraproducente porque tendrá problemas de integración con el equipo y fracasará en el marco de la individualidad.

La recomendación es que al momento de elegir los participantes de un equipo de trabajo para un proyecto de software, es muy importante evaluar no solamente sus conocimientos técnicos, sino que sus valores, principios y metas personales sean uniformes, de tal manera que encajen con el resto del equipo, y este equilibrio permita una sinergia que posibilite “ganar el campeonato”, aunque sus conocimientos individuales no sean los más completos[124].

En resumen: “**FORME UN EQUIPO UNIFORME**”

LECCIÓN 16:

Qué pasaría si es que un profesional vital para un proyecto de desarrollo de software, decide retirarse del mismo a medio camino?

Si el cerebro fuese una computadora, diríamos que sería necesario hacer una copia del conocimiento de la persona que se retira, trasladarlo en una memoria flash a la nueva persona, y restauraríamos ese conocimiento a su cerebro, de tal manera que continúe el nuevo profesional sin variaciones.

Obviamente que la tecnología actual no permite tal cosa, y más bien deja en evidencia que si una persona se retira a mitad del camino, inevitablemente se llevará consigo múltiples conocimientos que fue ganando en el tiempo.

[124] Hay que tener mucho en cuenta que es mejor formar un equipo uniforme y altamente comunicativo, pues si hay deficiencias de conocimiento, es fácil lograr que el personal lo adquiera.

Este planteamiento propone la reflexión sobre las posibles estrategias que permitan “almacenar” el conocimiento ganado, de tal manera que se evite una pausa en las actividades, y genere un retraso general en el cronograma del proyecto.

La estrategia que se ha analizado se ajusta más a la mitigación de este problema, aunque existen también otras válidas, es la de hacer que el líder del proyecto sea quien conoce con detalle los conocimientos ganados por cada uno de los profesionales, y sea él quien capacite y gestione el cambio de los profesionales. Mientras ese proceso se da, él asumirá ciertas partes de la programación que le facilite a la nueva persona el asumir el rol.

Si el cambio se da a nivel de gerente de proyecto, se debe contar con un desarrollador clave, que le sirva de “Backup” en sus conocimientos y estrategias aplicadas. Esto junto con una capacitación, harán que el cambio sea menos traumático.

Aún así, es imposible que se transmita toda la información. Es por ello que se deberá complementar la estrategia pidiéndoles a los desarrolladores que incluyan comentarios del negocio aplicado, dentro del código fuente del sistema. Esto le facilitará a cualquier persona nueva ubicarse más rápidamente en el proyecto.

En resumen: “**ADMINISTRE EL CONOCIMIENTO**”

Bibliografía

- Bruch, H., & Menges, J. (2010). La trampa de la aceleración. *Harvard Business Review América Latina* , 76-85.
- Carmel, E. (2000, Diciembre). *SOFTWARE DEVELOPMENT*. Retrieved Enero 25, 2010, from American University Washington DC.: http://www1.american.edu/carmel/hb2165a/software.htm
- Carnegie Mellon University. (2011). *CMMI*. Retrieved Abril 11, 2011, from Carnegie Mellon University: http://www.sei.cmu.edu/cmmi/
- Chambers & Asociated Pty Ltd. (2006, Julio 2). *Proxy Based Estimating*. Retrieved Abril 16, 2011, from C & A Software Engineers: http://www.chambers.com.au/Sample_p/co_proxy.htm
- Decretos Presidenciales, P. (2008, Abril 10). *Decretos Ejecutivos*. Retrieved Abril 1, 2011, from SIGOB: http://www.sigob.gob.ec/decretos/decretos.aspx?id=2007
- Dixon, Freeman, & Toman. (2011). No siga tratando de encantar a sus clientes. *Harvard Business Review* , 10.
- Eco-Finanzas. (n.d.). *Aversión al riesgo*. Retrieved Mayo 15, 2011, from Eco-Finanzas: http://www.eco-finanzas.com/diccionario/A/AVERSION_AL_RIESGO.htm
- Executive Brief. (2008). *Which Life Cycle Is Best for Your Project?* Retrieved Abril 12, 2011, from The Project Management Hut: http://www.pmhut.com/which-life-cycle-is-best-for-your-project
- Galorath, D. (2011). *SEER for Software Development*. Retrieved Abril 16, 2011, from Galorath - SEER: http://www.galorath.com/
- Hermans, M., Sioli, A., & Fay, P. (2010). Más allá del talento individual: Lecciones de Sudáfrica 2010. *Harvard Business Review* , 50-62.
- IBM. (2010). *Rational Unified Process*. Retrieved Abril 15, 2011, from IBM: http://www-01.ibm.com/software/awdtools/rup/?S_TACT=105AGY59&S_CMP=WIKI&ca=dtl-08rupsite
- Ing, K. (2010). No siga tratando de encantar a sus clientes. *Harvard Business Review América Latina* , 10.
- Liga Coop, A. (n.d.). *Cooperativismo*. Retrieved Abril 1, 2011, from Liga de Cooperativas de Puerto Rico: http://www.liga.coop/
- Lovallo, D., & Kahneman, D. (2010). La falsa ilusión del éxito. *Harvard Business Review* , 72-80.
- Martinez, S. (2009, Enero). Director Ejecutivo AESOFT.
- Microsoft. (2005). *MSF for CMMI Process Improvement*. Retrieved Abril 10, 2011, from MSF for CMMI Process Improvement: http://guides.brucejmack.biz/MSF%20for%20CMMI%20Process%20Improvement/Process%20Guidance/Supporting%20Files/Concepts.htm
- Microsoft. (2003). *Process Templates and Tools*. Retrieved Febrero 25, 2011, from Microsoft MSDN: http://www.microsoft.com/msf
- Moe, G. L. (2011). Caso de Estudio, Camino rápido al fracaso? *Harvard Business Review (América Latina)* , 89-93.

- Open Source Initiative, A. (2011). *Open Source Definition*. Retrieved Abril 1, 2011, from Open Source Initiative: http://www.opensource.org/docs/osd
- PRICE Systems. (2011). *Cost Estimating Tools for Software Projects*. Retrieved Abril 17, 2011, from TruePlanning by PRICE Systems: http://www.pricesystems.com/products/price_trueplanning.asp
- Process Strategies Inc. (2010, Abril 15). *Process Strategies: SCAMPI C Appraisal*. Retrieved Abril 10, 2011, from Process Strategies Inc.: http://www.process-strategies.com/SCAMPI_C.html
- Rowling, J. K. (2011). Recuperarse del Fracaso. *Harvard Business Review América Latina* , 75.
- Spolsky, J. (2007, Octubre 26). *Evidence Based Scheduling*. Retrieved Abril 17, 2011, from Joel on Software: http://www.joelonsoftware.com/items/2007/10/26.html
- Wikipedia, A. (2011, Abril 15). *Agile Software Development*. Retrieved Abril 15, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Agile_software_development
- Wikipedia, A. (2011, Abril 5). *Capability Maturity Model Integration*. Retrieved Abril 10, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration
- Wikipedia, A. (2011, Febrero 20). *Centralismo*. Retrieved Abril 1, 2011, from Wikipedia: http://es.wikipedia.org/wiki/Centralismo
- Wikipedia, A. (2011, Marzo 28). *Código Fuente*. Retrieved Marzo 28, 2011, from Wikipedia: http://es.wikipedia.org/wiki/C%C3%B3digo_fuente
- Wikipedia, A. (2011, Marzo 19). *Economía del don*. Retrieved Abril 1, 2011, from Wikipedia: http://es.wikipedia.org/wiki/Econom%C3%ADa_del_regalo
- Wikipedia, A. (2011, Abril 5). *Estimation Theory*. Retrieved Abril 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Estimation_theory
- Wikipedia, A. (2011, Marzo 3). *Extreme Programming*. Retrieved Marzo 15, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Extreme_Programming
- Wikipedia, A. (2011, February 16). *Freeware*. Retrieved Abril 1, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Freeware
- Wikipedia, A. (2011, Febrero 25). *Function Point*. Retrieved Abril 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Function_point_analysis
- Wikipedia, A. (2011, Marzo 23). *Infracción de derechos de autor*. Retrieved Abril 1, 2011, from Wikipedia: http://es.wikipedia.org/wiki/Infracci%C3%B3n_de_copyright
- Wikipedia, A. (2011, Marzo 28). *Lenguaje de Programación*. Retrieved Marzo 28, 2011, from Wikipedia: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n
- Wikipedia, A. (2011, Abril 11). *Monte Carlo method*. Retrieved Abril 17, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Monte_Carlo_method
- Wikipedia, A. (2011, Marzo 18). *Planning Poker*. Retrieved Abril 17, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Planning_poker

- Wikipedia, A. (2011, Junio 15). *PRICE Systems*. Retrieved Abril 17, 2011, from Wikipedia: http://en.wikipedia.org/wiki/PRICE_Systems
- Wikipedia, A. (2011, Marzo 8). *Program Evaluation and Review Technique*. Retrieved Abril 17, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Program_Evaluation_and_Review_Technique
- Wikipedia, A. (2011, Marzo 11). *Putnam Model*. Retrieved Abril 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Putnam_model
- Wikipedia, A. (2011, Abril 7). *Rayleigh Distribution*. Retrieved Abril 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Rayleigh_distribution
- Wikipedia, A. (2011, Abril 8). *Scrum (development)*. Retrieved Abril 15, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Scrum_(development)
- Wikipedia, A. (2011, Marzo 21). *SEER - SEM*. Retrieved Abril 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/SEER-SEM
- Wikipedia, A. (2011, Marzo 31). *Software Libre*. Retrieved Abril 1, 2011, from Wikipedia: http://es.wikipedia.org/wiki/Software_libre
- Wikipedia, A. (2011, Febrero 11). *V-Model*. Retrieved Abril 17, 2011, from Wikipedia: http://en.wikipedia.org/wiki/V-Model
- www.ISO15504.es. (2011, Abril 14). *Norma ISO 15504*. Retrieved Abril 15, 2011, from Portal de la norma ISO 15504 en castellano: http://www.iso15504.es/

Índice de Gráficos

Glosario de Términos

- Programación de Software: Es el proceso que permite trasladar un conjunto de modelos y estudios realizados sobre determinado problema, a código fuente sobre una herramienta informática de programación. El resultado final de la programación es un programa de computadora ejecutable.
- Arquitectura de Software: Es un proceso realizado por un experto en implementaciones tecnológicas y desarrollo de software, de tal manera que a través del planteamiento de escenarios y ponderaciones de priorización, se decide qué tipo de tecnología y estructura conceptual se utilizará a lo largo de todo un desarrollo de software.
- Lógica de Negocio: Cuando se desarrolla un software, se busca cumplir con un conjunto de requerimientos que normalmente son entregados por el cliente o usuario final del software, de tal manera que se cumpla con un objetivo en su empresa. A este conocimiento entregado por el cliente se le conoce como lógica de negocio, y será el que guíe la programación y pruebas.
- Escenario Especializado: Se refiere a que en la etapa de elaboración de una arquitectura de software, es necesario evaluar las condiciones específicas de la lógica de negocio y de la tecnología vigente, para realizar un planteamiento de un escenario de trabajo. A este conjunto de condiciones en un punto específico del tiempo se lo llama Escenario Especializado.
- QA: (Quality Assurance) es un término muy utilizado en el ámbito del desarrollo de software para referirse al proceso que implica realizar las

pruebas necesarias de las herramientas programadas, para asegurar la calidad de funcionamiento de un sistema.

- Código Fuente de una aplicación: El código fuente de un programa está escrito por un programador en algún lenguaje de programación, pero en este primer estado no es directamente ejecutable por la computadora, sino que debe ser traducido a otro lenguaje (el lenguaje máquina o código objeto) que sí pueda ser ejecutado por el hardware de la computadora. Para esta traducción se usan los llamados compiladores, ensambladores, intérpretes y otros sistemas de traducción. (Wikipedia, Código Fuente, 2011)
- Lenguaje de programación: Un lenguaje de programación es un idioma artificial diseñado para expresar algoritmos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (Wikipedia, Lenguaje de Programación, 2011)

Anexos

A continuación se presenta un listado de los anexos que se encontrarán en medio electrónico, junto con el presente documento:

Anexo 1	Base de datos de empresas de la AESOFT
Anexo 2	Código fuente de la aplicación desarrollada para ejecutar las encuestas
Anexo 3	Plantilla para la elaboración de una cotización
Anexo 4	Modelo de cálculo para obtener el tiempo resultante de un proyecto
Anexo 5	Plantillas para la elaboración de los documentos definiciones iniciales y de especificaciones funcionales