REVISTA TIA

Tecnología, Investigación y Academia -Red Avanzada – RITA
Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología

Competitividad desde las Metodologías de estimación y la gestión de los costos en la industria del software

Competitiveness from estimation methodologies and cost management in the software industry

Rodriguez Guzmán, Eder Javier[1]. Rodriguez Puentes, Zulma Yolima [2]. Álvarez Silva, Deyvid Leonardo[3]

Citar este documento:

Rodríguez-Guzmán, Eder Javier; Rodríguez-Puentes, Zulma Yolima; Álvarez-Silva, Deyvid Leonardo. Competitividad desde las Metodologías de estimación y la gestión de los costos en la industria del software. Revista Tecnol.Investig.Academia TIA, ISSN: 2344-8288, Vol. 10 Número 1, pp. 147- 171. Bogotá-Colombia.

Resumen

El dinamismo de la industria del software y la exigencia de ser más competentes en el mercado nacional e internacional hace necesario que se evalúen y replanteen la forma de estimar los costos de fabricación de software. En este artículo se presentan las diferentes metodologías usadas a la hora de estimar un proyecto de desarrollo con el fin de identificar los ítems asociados al 'Software o Hardware' que no son tenidos en cuenta pero que también inciden en los costos de un proyecto de desarrollo.

[1] Estudiante de especialización de Proyectos Informáticos, Universidad Distrital Francisco José de Caldas, Bogotá Colombia https://orcid.org/0000-0003-4298-1129
[2] Estudiante de especialización de Proyectos Informáticos, Universidad Distrital Francisco José de Caldas, Bogotá Colombia https://orcid.org/0000-0001-8751-8248
[3] Estudiante de especialización de Proyectos Informáticos, Universidad Distrital Francisco José de Caldas, Bogotá Colombia https://orcid.org/0000-0001-5844-770

Artículo de Reflexión Fecha de recepción: 11-12-2021. fecha Fecha de aceptación: marzo 18 2022 ISSN: 2344 8288 Vol. 10 No. 1

Por otra parte, también se habla acerca de las metodologías ágiles existentes que en los últimos años han sido una herramienta de apoyo para los proyectos de desarrollo, las cuales permiten mayor flexibilidad e inmediatez para moldearse al cambio del día a día de las necesidades del negocio y métodos de estimación. Para este caso puntual se busca identificar en Colombia qué metodología predomina en las organizaciones, y qué perspectiva se tiene de la forma de estimar en los proyectos de desarrollo de software con el fin de determinar la efectividad de los métodos actuales de estimación y que mejoras se pueden implementar
Palabras Clave: Estimar, Hardware, Metodologías Ágiles y Software.

Abstract

The dynamism of the software industry and the demand to be more competent in the national and international market make it necessary to evaluate and rethink the way to estimate software manufacturing costs. This article presents the different methodologies used when estimating a development project in order to identify the items associated with 'Software or Hardware' that are not taken into account and that also affect the costs of a development project.

On the other hand, it also talks about the existing agile methodologies that in recent years have been a support tool for development projects, which allow greater flexibility and immediacy to adapt to the day-to-day change of business needs. and estimation methods.

For this specific case, it is sought to identify in Colombia which methodology predominates in organizations, and what perspective is had of the way of estimating in software development projects in order to determine the effectiveness of current estimation methods and what improvements are made. can implement.

Key Words: Estimate, Hardware, Agile Methodologies and Software.

I. Introducción

A principios de los años 70, cuando la ingeniería de software era prácticamente inexistente. El término 'crisis del software' expresaba las dificultades del desarrollo de software frente al rápido crecimiento de la demanda de software, de la complejidad de los problemas a ser resueltos y de la inexistencia de técnicas establecidas para el desarrollo de sistemas que funcionan adecuadamente o pudieran ser validados. A Partir de la denominada "crisis del software" por Dijkstra en el discurso que pronunció durante la entrega del premio Turing en 1972 [1], se ha tratado de abordar mediante el planteamiento de nuevos métodos, metodologías, técnicas y paradigmas para minimizar su impacto. El alcance de dichas propuestas no se

limita exclusivamente a actividades relacionadas con el desarrollo en sí de los sistemas, sino que abarca también las actividades de gestión de estos [2]

Uno de los primeros pasos a realizar en un proyecto de software es la fase de planificación, en esta fase se encuentra una de las actividades más importantes de un proyecto 'la estimación', dicha estimación consiste en buscar un aproximado en tiempo de cuánto puede tardar en ejecutarse una tarea de acuerdo con su complejidad y tamaño. Al estimar los factores relacionados con un proyecto (esfuerzo, personal, cronograma, costo, etc.) Se requiere conocer o estimar su tamaño para evaluar las posibles soluciones, comparar alternativas y calcular costos antes de decidirse por un enfoque determinado [3].

Las estimaciones permiten evaluar en etapas tempranas la viabilidad económica del proyecto, no obstante, esta se ve limitada por la disponibilidad de información durante las primeras etapas del proyecto, por ello es importante el componente humano, ya que puede aportar experiencia y conocimiento. Por otra parte, pese a que las técnicas específicas para la estimación de costos predominan el uso de una sola, se observó que a partir del año 2010 ha aumentado el uso de técnicas híbridas, de acuerdo con estudios indican 56 formas de estimación para el año 2015 [4]. En el siguiente artículo se presenta la investigación sobre las metodologías ágiles usadas en las organizaciones de tecnología en Colombia y su perspectiva de los métodos usados de estimación en los proyectos de desarrollo de software.

II. Marco teórico

En el desarrollo de software es necesario tener en cuenta los principales factores de tiempo y coste, para identificar la mejor manera de producir un desarrollo es necesario conocer las diferentes metodologías que existen para estimar, algunas de ellas **son: a. Juicio de Expertos Esta** metodología se realiza a partir de la combinación de técnicas como lluvia de ideas, entrevistas y definición de tiempos bajo el criterio de expertos, los cuales son tenidos en cuenta para estimar el esfuerzo al desarrollar un proyecto. En la mayoría de las empresas donde se produce software para apoyar el negocio, las prácticas de estimación y planificación son débiles. En general, los administradores estiman el costo y la duración del proyecto a desarrollar utilizando solamente el juicio de un experto, lo que produce cronogramas y presupuestos poco acertados [5], sin tener en cuenta que unas buenas estimaciones conducen a tener una mayor visibilidad del estado del proyecto, mejoran la calidad de los productos entregados, aumentan la sintonía entre los atributos funcionales y no funcionales del software, entregan un presupuesto acertado y permiten tener relaciones de confianza entre las áreas técnicas y financieras del negocio [6].

b. Descomposición de tareas

También los proyectos se estiman por descomposición o EDT (Estructura de Desglose de Trabajo) en esta descomposición o desglose de trabajo se identifican los entregables y las tareas que se requieren para poder llegar a ellos, y así mediante una aproximación, se van sumando los esfuerzos requeridos hasta obtener el tiempo total necesario para desarrollar el proyecto. Una buena EDT provee un marco común para todos los entregables del proyecto y para tareas específicas dentro del proyecto. Por lo tanto, facilita la comunicación entre quienes están implementando el proyecto, lo que finalmente mejora la integración de los planes de tiempo, recurso y calidad del proyecto; una buena EDT al final produce mejores cronogramas y mejores estimaciones de costos [7].

Una EDT orientada a entregables define el trabajo del proyecto en términos de los componentes (físico o funcional) que componen el entregable. Una vez conocido el objetivo y el tipo de EDT a utilizar, el siguiente paso es crear la estructura, dotándola de los elementos necesarios para su completo desarrollo [8]. No existen reglas específicas para su diseño, pero en general, los elementos EDT deben cumplir los siguientes criterios: ● Debe ser creada con la ayuda del equipo.

- Definen y organizan la estructura de trabajo total del proyecto.
- Completan un nivel antes de seguir descomponiendo alguno de sus elementos.
- Subdividen el trabajo del proyecto en porciones más pequeñas, entendibles y fáciles de manejar. ● Cada nivel descendente representa una definición cada vez más detallada del trabajo del proyecto, hasta llegar al nivel más bajo o "paquete de trabajo".
- El trabajo comprendido en los paquetes de trabajo puede ser programado, presupuestado, controlado, y se le puede asignar un único responsable [8].

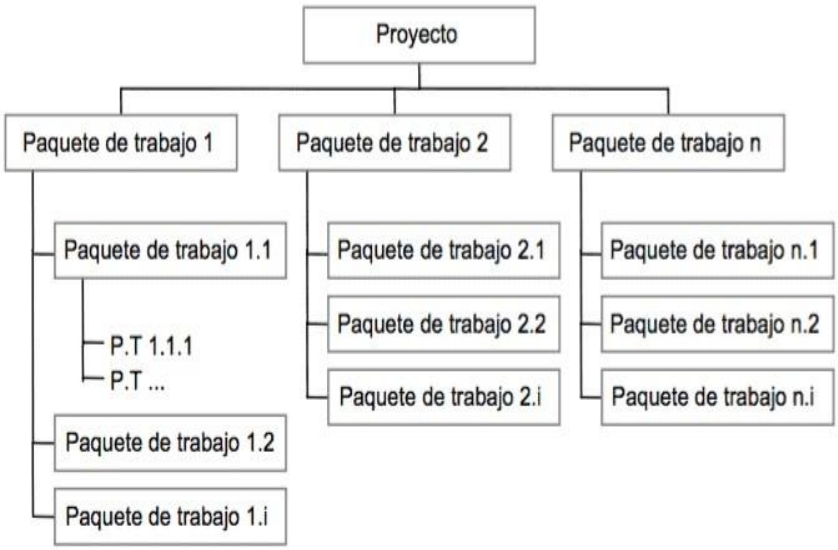

Figura 1. Descomposición de tareas de un proyecto [9]

c. Puntos de Casos de Uso

La metodología de los puntos de casos de uso es una derivación de la metodología de puntos de función propuesta por Albrecht. Karner, basa su metodología en la utilización de casos de uso como dato de entrada para calcular el esfuerzo en horas hombre (hh) que son necesarias para el desarrollo de un proyecto de software [10].

Este método de estimación de esfuerzo utiliza los actores y casos de uso relevados para calcular el esfuerzo que llevará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado [11].

Tabla 1. Criterios de puntos de caso de uso

Tipo de Caso de Uso	***Descripción***	***Factor de Peso***
Simple	*El caso de uso tiene de 1 a 3 transacciones*	*5*
	El caso de uso tiene de 4 a 7 transacciones	*10*
Medio Complejo	*El caso de uso tiene más de 8 transacciones*	*15*

El método de estimación del esfuerzo utiliza cuatro variables principales:

- **Clasificación de los Actores Involucrados**

Los actores involucrados en los casos de uso se clasifican de acuerdo con su característica intrínseca y la forma en que interactúan con el sistema. Un actor simple (peso=1) es aquel que representa una interfaz de programación o API (ej.: capa de abstracción); un actor medio (peso=2) es aquel que interactúa mediante un protocolo (ej.: TCP/IP, HTTP, FTP) y un actor complejo (peso=3) es aquel que interactúa por medio de una interfaz gráfica. A cada actor de acuerdo con esta clasificación le corresponde un valor el cual se denomina peso (El autor de la metodología no especifica el origen de los pesos en ninguno de los casos).

- **Clasificación de Caso de Uso**

Los casos de uso son clasificados de acuerdo con la cantidad de transacciones que poseen, incluyendo las transacciones de escenarios alternativos y excluyendo las extensiones o inclusiones de otros casos de uso. Un caso de uso simple (peso=5) es aquel que posee 3 o menos transacciones; uno medio (peso=10) es el que posee de 4 a 7 transacciones; y un caso de uso complejo (peso=15) es el que posee más de 7 transacciones.

Nuevamente, a cada caso de uso le corresponde un peso [10].

- **Factor de Complejidad Técnica del Proyecto de Software**

Los factores técnicos están definidos por las influencias técnicas que puedan afectar el proceso de desarrollo del sistema a construir. Cada factor técnico posee un grado de complejidad, que oscila entre 0 y 5, donde 0 significa un valor irrelevante o nulo y 5 determina un valor con alto grado de influencia. Cada factor técnico posee un valor de peso. El peso total de ese factor de influencia técnica se obtiene con el producto entre el

valor de complejidad asignado y el peso que le corresponde al factor.

- **Factores de Entorno del Proyecto**

Los factores de entorno indican la influencia del grupo humano involucrado en el proyecto sobre el sistema a desarrollar. De manera similar a los factores técnicos, los factores de entorno poseen un grado de influencia que oscila entre 0 y 5, donde 0 significa un valor irrelevante o nulo y 5 determina un valor con alto grado de influencia. Cada factor de entorno posee un valor de peso. El peso total de ese factor de influencia técnica se obtiene con el producto entre el valor de influencia asignado y el peso que le corresponde al factor de entorno [10].

d. PERT- ***Proyect o Program Evaluation and Review Technique***

PERT Significa *Proyect o Program Evaluation and Review Technique* (Técnica de evaluación y revisión de proyectos o programas) fue desarrollado a finales de la década de 1950 –1959 para planear y controlar los grandes proyectos de desarrollo armamentístico del ejército estadounidense.

Se creó para evidenciar la interdependencia de las tareas de los proyectos cuando se realiza la planificación de estos. En esencia, PERT es una técnica de modelos gráficos interrelacionados que consiste en pedir a los expertos el escenario pesimista, optimista y más probable, y mediante una fórmula matemática obtener el

valor esperado [12]. Los estimadores deben dar tres valores correspondientes al tiempo que puede llevar para ejecutar una tarea: (O) Optimista, (P) Pesimista y (M) Más probable.

Tabla 2. Fórmulas PERT [13]

Duración Esperada de la actividad	$EAD = (P + 4 + O)/6$
Desviación estándar de la actividad	$SP = (P - 0)/6$
Varianza de la actividad	$\boldsymbol{V = [(P - O)/6]2}$

Para cada actividad, se requiere estimar las siguientes cantidades: a = estimación de la duración de la actividad en las condiciones más favorables b = estimación de la duración de la actividad en las condiciones más desfavorables c = duración más probable de la actividad [14].

e. Punto Función

Puntos de Función (en inglés *Function Point Analysis*, FPA). El FPA fue introducido por Albrecht en 1970. Su propósito era solucionar algunos de los problemas asociados con el cálculo del tamaño del software en Líneas de Código (en inglés *Lines of Code*, LOC) y las medidas de productividad que se daban, especialmente por las diferencias en los cálculos de LOC que resultaban de los diferentes niveles de lenguajes que se utilizaban.

El interés de Albrecht para medir la funcionalidad del software desde el punto de vista del usuario, independientemente de la tecnología y lenguaje de programación, por lo que introdujo los Puntos de Función (PF) como una medida del tamaño de una aplicación desde el punto de vista funcional o del usuario [5]. Para aplicar esta metodología se debe identificar los requerimientos de software, los tipos de funciones esperadas (consultas, pantallas, campos de entrada y salida, entradas externas e internas, consultas a la base de datos, etc.) y la complejidad funcional de construir cada una de ellas (cuenta del número de requerimientos de cada tipo) luego dar un peso a la dificultad para desarrollar cada función y a partir de ellas calcular el esfuerzo requerido para cumplir con el proyecto [15].

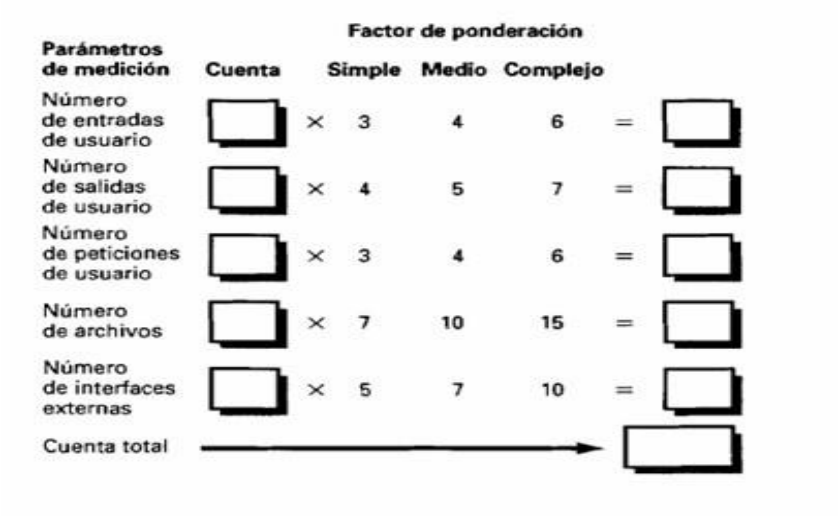

Figura 2. Factores de Ponderación [16]

Según el manual de IFPUG *(El International Function Point Users Group)* los objetivos del Análisis con Puntos de Función (APF) son los siguientes:

- Medir lo que el usuario solicitó y recibió.
- Medir en forma independiente de la tecnología usada para la implementación.
- Proporcionar una medida de tamaño para dar apoyo a análisis de productividad y de calidad ● Proporcionar un medio para la estimación del esfuerzo requerido para el desarrollo de software.
- Proporcionar un factor de normalización para realizar comparaciones de software [17].

f. COCOMO

El modelo COCOMO (*Costes Constructive Cost Model*) es uno de los sistemas de estimación de costes más utilizados en proyectos de desarrollo de software. Este tipo de proyectos se basa en tres submodelos: *básico, intermedio y detallado.*

Figura 3. Modelo COCOMO [18]

El modelo básico estima el coste del proyecto –pequeño o mediano- en función del número de líneas de código estimadas. En este modelo, el algoritmo COCOMO establece varios criterios de desarrollo, dependiendo el nivel de dificultad 'no del nivel de experiencia de los desarrolladores' sino de posibles dificultades que se pueden encontrar en el desarrollo o limitaciones del hardware usado en el desarrollo del software.

- El modelo intermedio se utiliza para estimaciones más complejas. Éste incluye 15 atributos –dentro de 4 categorías- del software para determinar el coste del proyecto. *Atributos del producto, Atributos del ordenador* usado, Atributos del personal y Atributos del proyecto. Todos estos atributos son ponderados matemáticamente atendiendo a su relevancia. De esta manera se intenta aproximar el coste estimado al real, lo máximo posible.

- El modelo detallado, incorpora las características del modelo intermedio y lleva a cabo una evaluación del impacto de los motivantes del coste en cada caso -análisis, diseño, etc.- del proceso de ingeniería del software [18].

Por otra parte, para un entorno competitivo donde el desarrollo es el motor que mueve la organización se hace necesario realizar una estimación en los costos y esto puede variar de acuerdo con las metodologías que implemente la organización, porque usar metodologías adecuadas permiten optimizar los tiempos en la entrega de productos y para esto las organizaciones emplean diferentes metodologías entorno al desarrollo, dentro de estas se encuentran las metodologías tradicionales y ágiles.

g. Metodologías Tradicionales

Las metodologías tradicionales son denominadas, a veces, de forma despectiva, como metodologías pesadas. Debido a que centran su atención en llevar una documentación exhaustiva de todo el proyecto, la planificación y control de este, en especificaciones precisas de requisitos y modelado y en cumplir con un plan de trabajo, definido todo esto, en la fase inicial del desarrollo del proyecto [19]. Para esto se inicia con la etapa de análisis y diseño, en donde el desarrollo del proyecto comienza a través de un estricto proceso de adquisición de requisitos, en donde se busca garantizar que los resultados sean de alta calidad en el producto final.

En las metodologías tradicionales se concibe un solo proyecto de grandes dimensiones y estructura definida; se sigue un proceso rígido que no cambiará durante la ejecución del proyecto; los requerimientos son acordados de una vez, demandando grandes plazos de planeación previa y poca comunicación con el cliente [20].

- **Metodología RUP**

La metodología RUP es conocida por su abreviatura en inglés de "*development method ó Unified Development Process* "es un proceso de desarrollo de software que, junto con el lenguaje de modelado unificado UML, constituye el método estándar más utilizado para el análisis, la implementación y la documentación de sistemas orientados a objetos [21].

El Proceso Unificado Racional es un proceso de producto, desarrollado y financiado por *Rational Software*, grupo que se encarga de trabajar de cerca con clientes y socios en busca del aseguramiento de que todo proceso sea actualizado y mejorado constantemente para evolucionar y probar mejores prácticas [22].

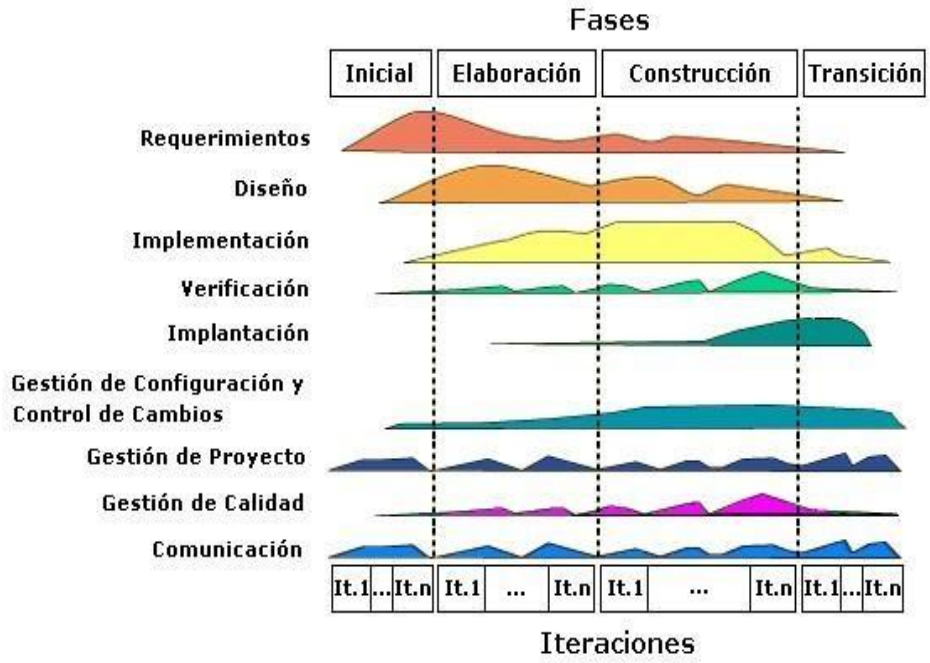

Figura 4. Fases Proceso Unificado Rational [23]

Las fases de la metodología RUP se compone de las siguientes:

○ *Iniciación o Diseño:* énfasis en el alcance del sistema; ○ *Preparación:* énfasis en la arquitectura; ○ *Construcción:* énfasis en el desarrollo; ○ *Transición:* énfasis en la aplicación.

RUP se basa también en las 4 Ps:

- Personas
- Diseño
- Producto
- Proceso

Las siguientes son consideradas las disciplinas de la metodología RUP

1. Modelado de negocios: Entiende los problemas e identifica mejoras potenciales, asegura que los participantes en este modelo tengan el entendimiento del problema, deriva los requerimientos del software.

2. Requerimientos: Mantiene a los interesados sobre lo que el proyecto debe realizar, define los límites y requerimientos, se enfoca en las necesidades del usuario y hace una base de costos.

3. Análisis y diseño: Transforma los requerimientos al diseño y su arquitectura robusta y lo adapta para corresponder al ambiente de implementación y ajustarla para un desempeño esperado.

4. Implementación: Define el código, convierte el diseño en archivos ejecutables, prueba los componentes desarrollados como unidades, integra esas unidades en un sistema ejecutable.

5. Pruebas: Se enfoca en la evaluación de la calidad del producto, encuentra las fallas y las documenta, valida los requerimientos planteados y el buen funcionamiento.

6. Transición: Describe las actividades entre el aseguramiento de la entrega y disponibilidad del producto hacia el usuario final, hay un énfasis entre probar el software en el sitio de desarrollo.

7. Administración y configuración del cambio: Consiste en controlar los cambios y mantiene la integridad de los productos que incluye el proyecto.

8. Administración de proyectos: Provee un marco de trabajo para administrar los proyectos, guías para la planeación, soporte y ejecución, un marco de trabajo para administrar los riesgos.

9. Ambiente: Se enfoca en las actividades para configurar el proceso del proyecto, describe las actividades requeridas para apoyar el proyecto, su propósito para proveer a las organizaciones de desarrollo de SW del ambiente necesario.[24]

Se destaca de esta metodología el acercamiento de las partes de forma estricta para la asignación tareas y roles además de responsabilidades dentro de una organización de desarrollo. Su principal objetivo es asegurar la producción de software que satisfaga los requerimientos que solicita el cliente.

La metodología es trabajada por un cronograma desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. (Customización). Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación [25].

- **Metodología MSF**

La metodología MSF o MICROSOFT SOLUTION FRAMEWORK (MSF) por sus siglas en inglés es una de las metodologías más frecuentadas en los procesos de desarrollo;

MSF comprende un conjunto de modelos, conceptos y guías que contribuyen a alinear los objetivos de negocio y tecnológicos, reducir los costos de la utilización de nuevas tecnologías, y asegurar el éxito en la implantación de las tecnologías Microsoft [26].

Los cinco modelos de MSF son:

1. Modelo de Arquitectura Empresarial de MSF
2. Modelo de Aplicaciones de MSF
3. Modelo de Equipos de Trabajo de MSF
4. Modelo de Procesos de MSF
5. Proceso de Diseño de Soluciones con Componentes
6.

Figura 5. Ciclo de vida MSF [26]

- **Metodologías Ágiles**

En principio para hablar de metodologías ágiles es necesario conocer el concepto de tradicional pues de ahí nace la metodología ágil.

En febrero de 2001, tras una reunión celebrada en Utah EEUU, nace el término “ágil” aplicado al desarrollo de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto [27] Tras esta reunión se creó The Agile Alliance [28], esta organización sin ánimo de lucro que se dedica a promover los conceptos relacionados con el desarrollo ágil de software, esta organización colabora a otras a adoptar los conceptos de metodologías ágiles. Por medio del Manifiesto Ágil, el cual es un documento que resume la filosofía "Agile", los principios del manifiesto ágil son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento [29].

Tabla 3. Comparación de metodologías Tradicionales vs Ágiles

Metodologías Tradicionales	***Metodologías Ágiles***
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Numerosas políticas	Proceso menos controlado
Existe un contrato prefijado	No existe un contrato tradicional
El cliente interactúa con el equipo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes y posiblemente distribuidos	Grupos pequeños menores a 10 integrantes
Más artefactos	Pocos artefactos
Más roles	Pocos roles
La arquitectura de software es fundamental	Menos énfasis en la arquitectura del software

Como se puede apreciar en la tabla evidenciamos que las metodologías ágiles, son más baratas en tiempo y recursos, obteniendo los mismos o mejores resultados en comparación con las metodologías tradicionales que pueden ser rígidas, poco flexibles y con más.

Las metodologías Ágiles en comparación con las tradicionales van enfocadas más a los trabajos en equipos no conformado principalmente por menos de 10 integrantes los cuales realizan sus procesos en la elaboración del producto por proceso a través de entregables y revisiones de productos para resolver sus fallas o incidencias antes de que el producto salga a producción lo que permite una buena adaptación en el entorno y un manejo adecuado de los roles en el equipo. Las metodologías tradicionales se destacan por sus elevados niveles de tareas, haciéndolo poco flexible al ser un proceso con un fuerte control por el cliente en la toma de decisiones que giran en torno al producto.

- **Metodología XP**

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los programadores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes [30]. El ciclo de vida ideal de XP consiste en seis fases [30]: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Figura 6. Ciclo de vida XP [30]

- **Metodología CRYSTAL**

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependen del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, *Crystal Clear* (3 a 8 miembros) y *Crystal Orange* (25 a 50 miembros) [31].

- **Metodología SCRUM**

Es un marco que permite el trabajo colaborativo para la elaboración de software, en donde se reduce la complejidad del desarrollo de productos para cumplir con los requerimientos del cliente. En Scrum inicialmente planean el contexto y un estimado amplio de la entrega. Después desarrollan el estimado de la entrega basándose en el desarrollo del ambiente del proyecto.

El proceso de ciclo de vida de Scrum reconoce que el proceso de desarrollo fundamental está completamente indefinido y usa mecanismos de control para perfeccionar la flexibilidad, manipular lo impredecible y el control del riesgo [32]. Siendo esta la metodología que predomina en las entidades del estado, como lo muestra una micro investigación realizada al personal de las TIC´s. Mediante encuesta virtual enviada a 36 de los principales CIO´s de las instituciones públicas de Colombia a la cual 19 dieron respuesta.

La pregunta: *¿Qué metodología predomina en su entidad para el proceso de desarrollo de software?*

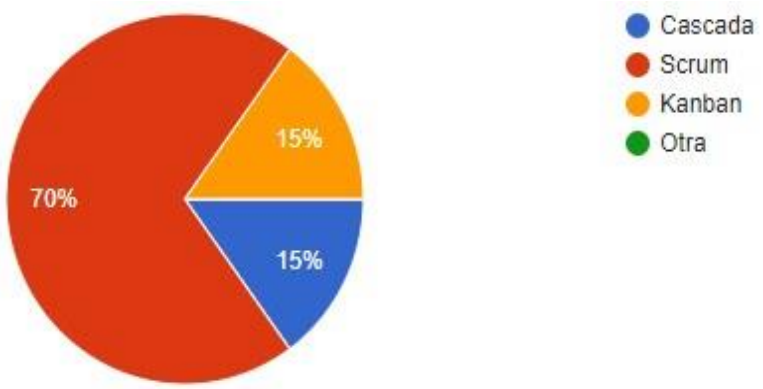

Figura 7. Metodología predominante en entidades

III. Contexto de la industria

a. Industria del software en Colombia

El mercado de software y tecnologías de la información de Colombia es el cuarto (4o) más grande de Latinoamérica, después de Brasil, México y Argentina. Durante los últimos 10 años el mercado de TI en Colombia ha crecido un 18%, el sector del software tuvo un crecimiento del 19.1% y en este mismo rango de tiempo los servicios de TI han crecido un 15.4% [33].

El 2020 fue un año crucial para la reafirmación de la industria del software, pues la crisis global generada por la pandemia se convirtió en una oportunidad de poner sobre la mesa lo fundamental y necesario que es para la modificación de modelos de negocios y la transición a la digitalización, lo que ha permitido que el software colombiano se afiance más como un producto de calidad para el desarrollo del país. [34]

Según el departamento nacional de estadísticas DANE en el año 2020pr el valor agregado del sector TIC ascendió a 35,1 billones de pesos presentando un decrecimiento de 1,1% con respecto a 2019p; mientras que en el año 2019p el valor agregado fue de 35,5 billones de pesos, con un crecimiento de 6,3% con respecto a 2018.[35] por lo que el sector informático y del desarrollo de software está creciendo cada día más. Mostrando que no se puede dejar de mencionar el papel importante y fundamental que juega la estimación en el momento de desarrollar un producto, porque de esto dependerá el ciclo de vida del desarrollo y el valor agregado que se podrá brindarle al cliente como lo es la calidad.

b. Perspectiva del cambio en la forma de estimar un proyecto de desarrollo de software

Se estima que un 70% de los fracasos expresados en atrasos y en la no estimación de costos reales asociados en los proyectos IT se debe a la no incorporación de prácticas esenciales y estandarización en los procesos de Software. No llevando a la práctica las acciones correctas, no tomando decisiones en el momento oportuno, y no ofreciendo compromiso. Uno de los mayores retos a los que se enfrentan las empresas hoy en día, es hacer que estas prácticas esenciales sean de su estrategia el centro de todas sus actividades. De esta forma la mayoría de las empresas IT mueren al cabo de 5 años.

Al principio en las organizaciones, el costo del software constituía un pequeño porcentaje del costo total de los sistemas basados en computadora. Un error considerable en las estimaciones del costo del software tenía relativamente poco impacto. Hoy en día, el software es el elemento más caro de la mayoría de los sistemas informáticos de las empresas. Un gran error en la estimación del costo puede ser lo que marque la diferencia entre beneficios y pérdidas [36]. Por ello y por el mundo cambiante en el que vivimos se hace necesario tener un cambio en la forma de estimar que permita minimizar errores y obtener mayores utilidades.

c. La gestión de costos como estrategia para ser competitivos

Un presupuesto, en economía, hace referencia a la cantidad de dinero que se necesita para hacer frente a cierto número de gastos necesarios para acometer un proyecto. De tal manera, se puede definir como una cifra anticipada que estima el coste que va a suponer la realización de dicho objetivo. [37]. Poder anticipar los recursos necesarios, el tiempo de entrega, la capacidad técnica del personal para poder desarrollar las

actividades y dar cumplimiento en las condiciones pactadas con los clientes y planeadas por las organizaciones generan la una mayor oportunidad de aumentar la rentabilidad de las compañías y de poder aumentar la competitividad.

Las metodologías de estimación como herramientas de presupuesto, al implementarse en el contexto colombiano por parte de las fábricas de software asociada a un concepto como el de Romain Haguenauer que implica que el concepto de competitividad es que también las empresas logren una mayor una participación en mercados internacionales según MINTIC entre los años 2015 a 2018 las exportaciones en las industrias de T.I. Alcanzaron a hasta US$ 231 Millones y en análisis de Procolombia las oportunidades de exportación en SOFTWARE de Agroindustrias, educación, energía, financiero, *retail*, salud y transporte, en destinos como Reino Unido, Estados Unidos, Canadá, México, entre otros.

Para analistas como Carlos Mallo Rodríguez una excelente gestión de costos inicia con la organización de la información, no solo para aumentar la competitividad, si no hacer uso de la información, estadísticas y datos almacenados para mantener la competitividad de estas empresas en el sector. No solo logrando ventajas competitivas, si no la implementación de un proceso de mejora continua. Según Michael Porter quien ha denominado el concepto costos al nivel de ser un área estratégica y su administración como la Gerencia Estratégica de Costos para desarrollar estrategias a efectos de alcanzar ventajas competitivas

sostenibles con una conjunción de tres elementos básicos 1, La cadena de valor. 2. El Posicionamiento Estratégico de la Empresa y 3. El de Causales de Costos para la Empresa.

El posicionamiento estratégico surge de tres fuentes, que no se excluyen entre sí, y a menudo se superponen:

- La producción es un subconjunto de los productos de una industria o servicio (Posición basada en la variedad).
- Atender las necesidades de un grupo particular de clientes (Posición basada en las necesidades), por ejemplo, el servicio personalizado de algunos bancos.
- Atender a un grupo de clientes en base a la accesibilidad de cualquier forma (Posición basada en el acceso), por ejemplo, por la proximidad geográfica. como lo menciona Ana María Golpe en La Gerencia Estratégica de Costos (G.E.C.) y los Costos de Transacción.

d. La Estimación de Costos en Instituciones Públicas Colombianas

Para Fedesof la industria del software en Colombia es altamente competitiva como lo indica en su estudio el Observatorio de Competitividad, Pero en contraste en una pequeña encuesta realizada a algunos CIO´s del sector estatal, la principal metodología de costeo es el juicio de expertos

La pregunta que se hizo fue: *¿Qué modelo de estimación de costos o presupuesto es más común en su entidad para los proyectos de desarrollo de software?*

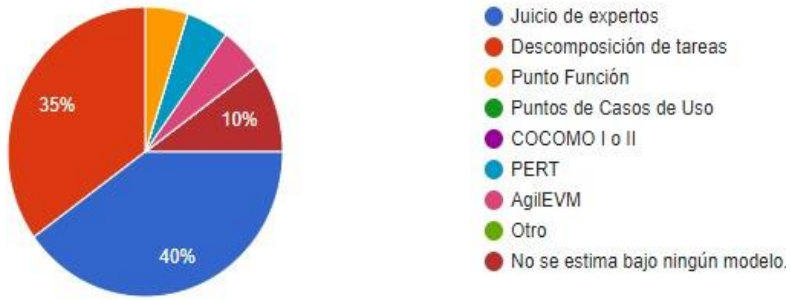

Figura 8. Pregunta 2

En la mayoría de los casos no se cumple con los tiempos estimados en los procesos de desarrollo y los presupuestos estimados para los diferentes proyectos. Como lo evidencia la encuesta.

La pregunta que se hizo fue: ¿Considera que cumplen los tiempos y presupuestos en el proceso de desarrollo de *software* de su entidad*?* Situación que se presenta en la mayoría de los casos, en el sector público.

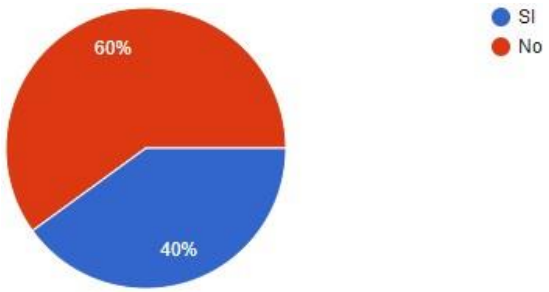

Figura 9. Pregunta 3

En conversación con algunos CIOS´s de entidades estatales muchas de las dificultades que se presentan en el desarrollo de software por parte de las entidades estatales en sus procesos de desarrollo para cubrir las diferentes necesidades a la medida que requieren y cumplir con el fin que se les asigna, son entre otras:

- Los cambios en la legislación que las regula.
- Inconsistencias en los requerimientos.
- Altas expectativas en las posibles soluciones de los sistemas a implementar.
- La burocracia en las diferentes etapas que hacen parte del proceso de desarrollo.
- No se aplica una metodología de estimación en la mayoría de los casos.

- Se pierde el control de los tiempos de desarrollo por interrupción en el proceso de desarrollo por cuestiones de contratación de personal.

Se evidenció en la micro investigación que no se tiene un proceso estandarizado para la determinación de los costos de fabricación de software, el elemento que predomina es el factor humano, sea por el valor del personal contratado por prestación de servicios o de vinculación asignado a los proyectos.

La pregunta que se hizo fue: *¿Qué elemento predomina en el proceso de evaluación de costeo de desarrollos internos de software en su organización?*

Figura 10. Pregunta 4

IV. Conclusiones

Colombia es uno de los países líderes en América latina en la producción de software y desarrollo en la industria de las TIC, poder implementar metodologías de costeo y de presupuestos, puede ser la herramienta para lograr atraer inversionistas y aliados desde el exterior, dar un plus de calidad a la industria de software en Colombia, tiempos más cortos en las diferentes etapas de la producción de software, apoyar el proceso de toma de decisiones en el nivel directivo.

La importancia en la selección de un modelo de estimación a utilizar varía de acuerdo con las características propias del proyecto, a la experiencia y apreciación de los responsables de liderar el proyecto.

El proceso de estimar bajo una metodología adecuada en los diferentes proyectos de desarrollo es una práctica que debe promoverse entre las empresas que se dedican a la industria del software como camino a la optimización de costos en los presupuestos asignados a cada proyecto.

V. Referencias

[1] Escola Tècnica Superior 'Enginyeria Informàtica, 2011, La Crisis del Software, https://histinf.blogs.upv.es/2011/01/04/la-crisis-del-software/

[2] J. A. I. P. Pow-Sang Portillo, Ricardo, (2005) “Estimación y Planificación de Proyectos Software con Ciclo de VidaIterativo-Incremental y empleo de Casos de Uso,” (p. 1).

[3] Fillottrani, P. R. (2007). Calidad en el Desarrollo de Software. Métricas de procesos de software

[4] H. G. Rodriguez Gutiérrez y P. A. Rojas Cubides, 2015, Técnicas de Estimación de Costos para Proyectos: Revisión Bibliográfica del 2005 al 2015

[5] Salazar B, G.,2009, p. 124 -125 -127). Estimación de proyectos de software: un caso práctico. Revista Ingeniería y Ciencia de la Universidad EAFIT, 5(9), 123-143. doi: https://publicaciones.eafit.edu.co/index.php/ingciencia/article/view/470/437

[6] Zapata & Chaudron, (2012). An Analysis of Accuracy and Learning in Software Project Estimating. 39th Euromicro Conference on Software Engineering and Advanced Applications, 414-421.
doi: http://doi.ieeecomputersociety.org/10.1109/SEAA.2012.46

[7] (Rad, P. F. & Cioffi, D. F. 2004, p. 6-7). Work and Resource Breakdown Structures for Formalized Bottom-Up Estimating. Cost Engineering, 46(2) https://www.researchgate.net/publication/237572249_Work_and_Resource_Breakdown_Structures_for_For
malized_Bottom-Up_Estimating

[8] M. Silvelo, 2016, RIB Spain, Estructura de Desglose del Trabajo EDT https://topodata.com/wpcontent/uploads/2019/10/EDT-Estructura-de-Desglose-del-Trabajo.pdf

[9] Estructura de Descomposición de Trabajo (WBS – Work Breakdown Structure) https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/planificacion/wbs/

[10] C. A. Remon, 2010, Análisis de Estimación de Esfuerzo aplicando Puntos de Caso de Uso, Instituto de
Investigación en Informática https://core.ac.uk/download/pdf/296343864.pdf

[11] (Estrada Molina. Escalona Griff L.,2010, p 2-3) Revista avanzada científica IDICT, Universidad de las Ciencias Informáticas. Cuba https://dialnet.unirioja.es/descarga/articulo/5074446.pdf

[12] (Aparicio GiL C., 2012) https://www.eoi.es/blogs/cesaraparicio/2012/05/06/el-modelo-cocomo-paraestimar-costes-en-un-proyecto-de-software/

[13] Lic. Miguel Cano PERT & CPM Investigación Operativa II

[14] UTFSM, 2004, Fundamentos de Investigaci´on de Operaciones CPM y PERT https://www.inf.utfsm.cl/~esaez/fio/s1_2004/apuntes/pert-2004-1.pdf

[15] (Jingchun, Fujie, Dandan & Fe, 2012, p. 122-135) https://www.ccsenet.org/journal/index.php/cis/article/view/16599

[16] Métricas de Puntos de Función http://planificacion-epis.blogspot.com/p/puntos-de-funcion.html

[17] J. Conrado, 1993, Puntos Funcional https://www.researchgate.net/publication/291695346_Puntos_de_Funcion

[18] (Aparicio Gil C, 2012). https://www.eoi.es/blogs/cesaraparicio/2012/05/06/el-modelo-cocomo-paraestimar-costes-en-un-proyecto-de-software/

[19] Maida, EG, Pacienzia, J. Metodologías de desarrollo de software. Universidad Católica Argentina, 2015. Disponible en: https://repositorio.uca.edu.ar/handle/123456789/522

[20] Khurana, H. & Sohal, J.S. Agile: The necessitate of contemporary software developers. International Journal of Engineering Science & Technology, 3(2), 1031-1039, 2011.

[21] Programa en línea Requerimientos.proceso-unificado-rational-rup línea 2021. Disponible en: https://www.programaenlinea.net/proceso-unificado-rational-rup

[22] Wendy Jaramillo W. Aplicación de la metodología RUP y el patrón de diseño MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel De La Guarda. /http://repositorio.puce.edu.ec/, 2016

[23] Ingeniería en Software whitesofft.blogspot. Metodologia RUP., 2017. Disponible en: http://whitesofft.blogspot.com/2017/10/metodologia-rup-la-metodologia-rup.html

[24] Requerimientos. línea 2021. https://www.fing.edu.uy/inco/cursos/ingsoft/pis/proceso/MUM/dat/intro/intro.htm

[25] Roberth G. Figueroa, Camilo J. Solís Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES https://www.researchgate.net/, 2008

[26] Practica 2: Metodologías Tradicionales https://sites.google.com/site/aessl13g314/practica-2/2-1/, 2014

[27] Boehm, B. Turner, R. Balancing Agility and discipline. A guide for the Perplexed. Addison-Wesley. 2003.

[28] Cockbun, A. Agile Software Development.. Addison-Wesley. 2001.

[29] Canós, J., Letelier, P. y Penadés, M. Metodologías Ágiles en el desarrollo de Software. Universidad Politécnica de Valencia, Valencia, 2003

[30] Beck, K. “Extreme Programming Explained. Embrace Change”, Pearson Education, 1999. Traducido al español como: “Una explicación de la programación extrema. Aceptar el cambio”, Addison Wesley, 2000.

[31] Fowler, M., Beck, K., Brant, J. “Refactoring: Improving the Design of Existing Code”. Addison-Wesley. 1999

[32] Schwaber, k. Agile Project Management with Scrum (Microsoft Professional). , 2004.

[33] Federación Colombiana de la Industria del Software y Tecnologías Informática. http://www.fedesoft.org Julio, 2019

[34] RED FORBES Industria del software: un sector con visión de crecimiento, 2021

[35] DANE,Cuenta Satélite de las Tecnologías de la Información y las Comunicaciones (CSTIC) https://www.dane.gov.co/index.php/, 2021

[36] (Alejandro Bedini González, p 54-55) libro Gestión de Proyectos de software https://www.inf.utfsm.cl/~guerra/publicaciones/Gestion%20de%20Proyectos%20de%20Software.pdf

Publicación Facultadde Ingenieríay Red de Investigacionesde TecnologíaAvanzada_ RITA

REVISTA

TIA