UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA DE SISTEMAS – PLANEACIÓN

EVALUAR DOS ALTERNATIVAS PARA DETERMINAR EL MEJOR RENDIMIENTO DE UNA FÁBRICA DE SOFTWARE:

UN CASO DEL SECTOR BANCARIO

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

LAURA REYES ORTEGA

TUTOR PRINCIPAL

GABRIEL DE LAS NIEVES SÁNCHEZ GUERRERO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

CIUDAD UNIVERSITARIA, CD. MX. NOVIEMBRE 2020

JURADO ASIGNADO:

Presidente:	Dr. Suárez Rocha Javier
Secretario:	Dr. Sánchez Lara Benito
Vocal:	Dr. Sánchez Guerrero Gabriel D.
1er. Suplente:	M. I. Soler Anguiano Francisca Irene
2do. Suplente:	Dr. Rivera Colmenero José Antonio

CIUDAD DE MÉXICO

TUTOR DE TESIS:

GABRIEL DE LAS NIEVES SÁNCHEZ GUERRERO

_____

FIRMA

Agradecimientos

A la vida que en estos tiempos históricos de pandemia me brinda el privilegio de contar con mis padres Rocío Ortega González y Manuel Reyes Valdez que me apoyan en todas aquellas ideas que se me ocurren, como esta, de continuar estudiando.

A mis hermanas Lety y Ximenita por escucharme y alentarme en los momentos difíciles, por estar ahí cuando más las necesito.

A mi director de tesis Dr. Gabriel De Las Nieves Sánchez Guerrero, que conocí desde el inicio de mis estudios de posgrados tomando los cursos propedéuticos que impartía, por aceptar trabajar juntos en esta investigación aún cuando había solicitado su año sabático y sería un reto la guía virtual.

A mi jurado asignado: Dr. Suárez Rocha Javier, Dr. Sánchez Lara Benito, M. I. Soler Anguiano Francisca Irene y Dr. Rivera Colmenero José Antonio por tomarse el tiempo de revisar y proponer mejoras para que esta tesis se hiciera realidad.

Reitero mis agradecimientos como en mi tesis del 2014: familia, amigos y profesores: ni el tiempo que me resta de vida será suficiente para agradecer todo lo que han hecho por mí.

“Cualquier momento es perfecto para aprender algo nuevo.”

Albert Einstein

Laura Reyes Ortega

lareortega@hotmail.com

Contenido

Índice de tablas

Índice de ilustraciones

Resumen

La presente investigación evalúa dos alternativas para organizar el área de desarrollo de software de un banco mexicano.

La primera alternativa (alternativa A) es la organización actual, donde un equipo de personas se encarga de llevar a cabo todas las actividades del proceso de desarrollo de software.

Sin embargo, al observarse una serie de problemáticas con este modelo, nos volteamos a ver cómo se organizan algunas áreas de desarrollo externas y encontramos que la mayoría de ellas se organizan mediante lo que llamamos la alternativa B, donde existen varios equipos dedicados a cada una de las actividades del proceso de software.

Para evaluar dichas alternativas nos basamos en directrices y estándares recientes y globales como el Modelo de Madurez de Capacidad Integrada (CMMI), además de ello queremos ver si la alternativa B es operacionalmente viable para el área.

Palabras clave: Evaluación, Nivel de madurez, Proceso de desarrollo de software.

Abstract

This research evaluates two alternatives for organizing a Mexican bank´s software development area.

The first alternative (alternative A) is the current organization, where a team of people is responsible for carrying out all the activities of the software development process.

However, when we look at a number of problems with this model, we turn to see how some external development areas are organized and we found that most of them are organized by what we call alternative B, where there are several teams dedicated to each of the activities of the software process.

In order to evaluate these alternatives we rely on recent and global guidelines and standards such as the Integrated Capacity Maturity Model (CMMI), in addition to this we want to see if alternative B is operationally viable for the area.

Keywords: Evaluation, Maturity Level, Software Development Process.

1. Introducción

Problemática

Debido a la creciente necesidad de crear software de manera rápida y con el mínimo de errores, es necesario contar con un área de desarrollo y mantenimiento de software que responda de manera eficiente a la operación de los distintos procesos de la actividad del banco.

Actualmente la fábrica de software cuenta con ingenieros encargados de llevar a cabo todas las etapas del desarrollo de sistemas, es decir, un mismo equipo es encargado de realizar el análisis, el desarrollo, la puesta a producción y el mantenimiento de un sistema.

Esta situación genera retrasos en las fechas de entrega comprometidas a las áreas usuarias pues los incidentes de las aplicaciones que ya se encuentran operando suelen ocurrir de manera inesperada y se deben atender de manera prioritaria, pues en muchos casos detienen los procesos del área usuaria. Siendo el mismo equipo el encargado de dar solución a estos incidentes y a la vez encargado de desarrollar las aplicaciones nuevas, a una de estas dos tareas se le debe dar prioridad.

Sumado a ello, la cantidad de errores que presentan las aplicaciones en ambiente productivo no es controlada. Debido a que el equipo que desarrolla el código también es el encargado de realizar las pruebas al sistema, muchas veces las pruebas se realizan sobre el escenario por default y no considerando escenarios alternativos, aplicando técnicas de pruebas automáticas o con la visión de un hacker.

Por otro lado, las tecnologías y técnicas que se utilizan al desarrollar el código de las aplicaciones son elegidas con base en la experiencia del desarrollador y no tomando en cuenta las necesidades específicas del proyecto. Provocando que cuando esa persona se va del área (ya sea por movilidad, jubilación, incapacidad, etc.) al nuevo analista le toma bastante tiempo entender su arquitectura para su posterior mantenimiento.

Tras voltear a ver como desarrollan software otras organizaciones, se ha encontrado que tienen un esquema en donde se cuenta con un equipo dedicado a cada una de las etapas, es decir, un equipo encargado de analizar el proyecto, otro equipo enfocado a codificar, uno más dedicado a probar los desarrollos y finalmente uno enfocado a dar mantenimiento.

Es por ello que se concluye con la necesidad de comparar la manera actual con la que se crea el software y la manera de hacerlo en equipos de trabajo segmentados para saber si un esquema así conviene para la fábrica de software del banco, dada sus características particulares de tamaño, recursos, personal y usuarios.

Objetivo
Mejorar el rendimiento de una fábrica de software.

Meta
Ofrecer la alternativa que mejora el rendimiento de una fábrica de software.

Importancia para abordarlo
La calidad de un producto de software, definida como la capacidad del software para permitirles a usuarios específicos lograr las metas propuestas con eficacia, productividad, seguridad y satisfacción[1], está estrechamente relacionada por la calidad del proceso con el que se desarrolla.

A medida que se mejora el proceso, también incrementa la calidad de los sistemas.

Detectar los errores de manera temprana, a diferencia de encontrarlo cuando el sistema ya se encuentra en uso, requerirá menos tiempo para su depuración.

2. Estado del arte
A continuación, se pretende tener un panorama de los estudios que se han realizado sobre las formas de organizar una fábrica de software, en concreto cuales son las formas más comunes, los principales obstáculos que se han encontrado y finalmente que tendencias hay hacia el futuro.

Formas de organización de fábricas de software más comunes
Las alternativas de organización comúnmente encontradas en los departamentos de desarrollo de software son:

- Organizaciones centradas en proyectos
- Organizaciones centradas en el departamento.
- Organizaciones matriciales
- Organizaciones de línea de productos.

Organizaciones centradas en proyectos

Son típicamente vistas en organizaciones pequeñas o recién formadas. Es adecuada para grupos de aproximadamente 5 a 40 personas que apoyan de 1 a 8 proyectos de duración pequeña a mediana, quizás hasta un año cada uno.

Cada grupo es principalmente autosuficiente y cuenta con suficientes desarrolladores capacitados para abordar cada etapa del ciclo de vida del desarrollo. Esto a su vez significa que la mayoría de las personas tendrán la responsabilidad de alguna faceta del desarrollo que no sea solo la programación, como los requisitos, la arquitectura, la administración de la configuración o las pruebas.

A medida que las organizaciones de desarrollo crecen, las organizaciones centradas en proyectos se vuelven menos deseables.

Organizaciones de desarrollo centradas en el departamento

Comienzan a ser prácticas a medida que un grupo crece por encima de 25 desarrolladores o 5 proyectos. En estos niveles de personal, hay suficientes personas para formar múltiples departamentos centrados en habilidades de software particulares o áreas del ciclo de vida. Por ejemplo, un grupo de 40 personas podría tener departamentos para:

- Administradores de sistemas y bases de datos.
- Programadores de interfaz de usuario.
- Programadores de aplicaciones.
- Gestión de configuración, prueba y garantía de calidad.

Organizaciones matriciales

Cuando la organización crece a varios cientos de personas o más, es posible que desee considerar una organización matricial.
Las organizaciones matriciales a veces se utilizan en empresas con una gran cantidad de desarrolladores de software que trabajan en una amplia gama de proyectos de software. Un lado de la matriz se organiza a lo largo de conjuntos de habilidades, mientras que el otro lado de la matriz se organiza a través de proyectos.
En una organización matricial, cada desarrollador tiene dos gerentes. Un gerente es del departamento o matriz de conjunto de habilidades y un gerente es de la matriz del proyecto. Un desarrollador generalmente permanece en un solo departamento mientras él o ella continúen trabajando en esa área de habilidades. Un desarrollador solo permanecería en un proyecto durante el tiempo que se necesitara su habilidad particular y luego regresaría a su departamento para otra tarea.

Organizaciones de línea de producto

Los desarrolladores se organizan en proyectos basados en líneas de productos de negocio en lugar de departamentos de conjunto de habilidades.
Una organización de línea de productos es responsable de contratar los conjuntos de habilidades requeridos para su combinación de proyectos.
Por ejemplo, una línea de productos puede requerir analistas de requerimientos, expertos en sistemas operativos, algunos desarrolladores web y gestión de configuración. Otra línea de productos podría tener analistas de requerimientos, expertos en codificación en tiempo real y gestión de la configuración.

Esto funciona si las organizaciones de la línea de productos son lo suficientemente grandes como para que existan suficientes desarrolladores para el personal de funciones duplicadas en todos los departamentos.
La desventaja es que los proyectos de software a menudo requieren diferentes conjuntos de niveles de habilidad en diferentes momentos del ciclo de vida del software. Cada línea de productos siempre debe tener suficientes recursos para el personal durante los períodos pico mientras se preocupa por las pausas intermedias.

Principales oportunidades en el desarrollo de software

Dividir a los arquitectos de software en un departamento o grupo separado

Se ha descubierto que esto puede conducir al elitismo y ser muy contraproducente. Primero, comienza a separar a los arquitectos de los desarrolladores que están haciendo la implementación real. Los arquitectos se desconectan más rápidamente con las últimas metodologías de desarrollo que realmente se utilizan.

Combinar el desarrollo de software y operaciones en un solo equipo

El trabajo de las operaciones es mantener las aplicaciones en funcionamiento. La forma más fácil de hacer esto es nunca cambiar nada. La combinación de desarrollo y operaciones en un solo equipo tiene la tendencia natural de sofocar la innovación. Las organizaciones de desarrollo de software deben separarse de las operaciones para permitir que se desarrollen aplicaciones nuevas y modificadas según sea necesario para satisfacer las necesidades de negocio.

Separación de grupos de desarrollo de software y mantenimiento de software.

Cuando se separan los grupos de desarrollo y mantenimiento de software, termina requiriendo los mismos tipos de especialistas en software para cada grupo. Esto significa que tiene que duplicar el número de empleados en su organización para obtener especialistas o forzar a un grupo a reducir la especialización. Además, se hace más difícil crear incentivos para que los desarrolladores de software hagan las cosas bien la primera vez, ya que saben que otro grupo será responsable de corregir los errores.[2]

Tendencias en el desarrollo de software

Entre ahora y el 2025, la capacidad de las organizaciones, sus productos, sistemas y servicios para competir, adaptarse y sobrevivir dependerá cada vez más del software.[3]

Tendencias de los sistemas y su influencia en los procesos de ingeniería de software y sistemas:

1. La creciente integración de la ingeniería de software y la ingeniería de sistemas
2. Un mayor énfasis en los usuarios y el valor final
3. Criticidad de software y sistemas y tendencias de confiabilidad
4. Cambio cada vez más rápido
5. Tendencias de globalización e interoperabilidad
6. Sistemas cada vez más complejos
7. Necesidades crecientes de COTS (alternativas a desarrollos caseros), reutilización e integración heredada
8. Abundancia computacional.

La creciente integración de la ingeniería de software y la ingeniería de sistemas

Varias tendencias han hecho que la ingeniería de sistemas y la ingeniería de software evolucionen como procesos en gran medida secuenciales e independientes. Primero, la ingeniería de sistemas comenzó como una disciplina para determinar la mejor manera de configurar varios componentes de hardware en sistemas físicos como barcos, ferrocarriles o sistemas de defensa.

En segundo lugar, hacer que personas que no son de software determinen las especificaciones del software a menudo hace que el software sea mucho más difícil de producir, colocando el software aún más prominente en la ruta crítica del desarrollo del sistema. Elegirían los mejores componentes del sistema cuyo software era incompatible y requería mucho tiempo integrarlo.

Las directrices y estándares recientes del proceso, como el Modelo de madurez de capacidad integrada (CMMI), ISO / IEC 12207 para ingeniería de software e ISO / IEC 15288 para ingeniería de sistemas enfatizan la necesidad de integrar sistemas e ingeniería de software procesos. Enfatizan prácticas tales como la ingeniería concurrente de requisitos y soluciones, el desarrollo integrado de productos y procesos, y procesos basados en riesgos versus procesos basados en documentos.

Tendencias de énfasis de usuario / valor e implicaciones de proceso

La usabilidad y los costos totales, incluidos la ineficiencia del usuario y los costos de ineficacia, se están convirtiendo en las principales prioridades de las organizaciones de usuarios de TI. Esto se refleja cada vez más en las actividades de selección de productos de los usuarios, con criterios de evaluación que enfatizan cada vez más la usabilidad del producto y el valor agregado frente a un fuerte énfasis previo en las características del producto y los costos de compra.

Dichas tendencias afectarán en última instancia las prioridades de productos y procesos de los productores, las estrategias de marketing y la supervivencia competitiva.

Se necesitan modelos más adaptativos e impulsados por el riesgo. Más fundamentalmente, la teoría que subyace a los modelos de proceso de software debe evolucionar desde una visión del mundo puramente reduccionista "moderna" (universal, general, atemporal, escrita) a una síntesis de estas y visiones del mundo situacionales "posmodernas" (particular, local, oportuna, oral).

Criticidad de software y sistemas y tendencias de confiabilidad

Aunque la dependencia del software por parte de las personas, sistemas y organizaciones se está volviendo cada vez más crítica, la fiabilidad generalmente no es la máxima prioridad para los productores de software. Es probable que esta situación continúe hasta que una catástrofe importante de sistemas inducida por software tenga un impacto similar en la conciencia mundial que la catástrofe del World Trade Center del 11 de septiembre que estimule la acción para establecer la responsabilidad por la confiabilidad del software.

Las estrategias de proceso para sistemas intensivos en software altamente confiables y muchas de las técnicas para abordar sus desafíos han estado disponibles durante bastante tiempo. Una conferencia histórica de 1975 sobre software confiable incluyó documentos sobre especificación formal y procesos de verificación, eliminación temprana de errores, tolerancia a fallos, árbol de fallas y modos de falla y análisis de efectos, teoría de pruebas, procesos y herramientas, verificación y validación independiente, análisis de causa raíz de datos empíricos, y el uso de ayudas automatizadas para la detección de errores en las especificaciones y códigos de software.

Tendencias de cambio rápido

El ritmo cada vez más rápido del cambio de TI está impulsado por tendencias tecnológicas como la Ley de Gordon Moore (la densidad y el rendimiento del transistor se duplica aproximadamente cada 18 meses), además de la continua necesidad de diferenciación de productos y procesos similares a los tornados para la introducción de nuevas tecnologías. La conectividad global también acelera los efectos de la tecnología, el mercado y los cambios tecnológicos. El cambio rápido también aumenta la prioridad de la velocidad de desarrollo frente al costo de capitalizar las ventanas del mercado.

Las necesidades de alcanzar altos niveles de agilidad presentan un desafío similar al mismo tiempo que se logra el alto nivel de disciplina necesario para desarrollar

sistemas altamente confiables. Implica identificar las partes de la aplicación que más necesitan agilidad y encapsularlas en un marco basado en un plan. Ejemplos frecuentes de partes del sistema ágil encapsulado son las partes que se ocupan de cambios impredecibles en las interfaces de usuario o las interfaces externas del sistema.

Tendencias de globalización e interoperabilidad

La conectividad global proporcionada por Internet proporciona grandes economías de escala y economías de red que impulsan las estrategias de producto y proceso de una organización. Los servicios de distribución y movilidad independientes de la ubicación crean nuevas bases ricas para la colaboración sinérgica y desafíos en las actividades de sincronización. Los salarios diferenciales brindan oportunidades para ahorrar costos a través de la subcontratación global, aunque la falta de una preparación cuidadosa puede convertir fácilmente los ahorros en excesos. La capacidad de desarrollarse en múltiples zonas horarias crea la posibilidad de un desarrollo muy rápido a través de operaciones de tres turnos, aunque nuevamente hay desafíos significativos en la visibilidad y el control de la administración, la semántica de la comunicación y la construcción de valores y confianza compartidos.

Se necesita mucho trabajo para establecer patrones sólidos de éxito para los procesos de colaboración global. Los desafíos clave incluyen puentes transculturales; establecimiento de una visión común compartida y confianza; mecanismos de contratación e incentivos; traspasos y sincronización de cambios en el desarrollo de zonas horarias múltiples. La mayoría de los paquetes de software están orientados al uso individual; solo determinar la mejor manera de apoyar a los grupos requerirá una gran cantidad de investigación y experimentación.

Tendencia de sistemas complejos

Durante la década de 1990 y principios de 2000, comenzaron a surgir normas como ISO / IEC 12207 e ISO / IEC 15288 que ubicaban los sistemas y procesos de proyectos de software dentro de un marco empresarial. Al mismo tiempo, arquitecturas empresariales como el IBM Zachman Framework, RM-ODP y el U.S.Federal Enterprise Framework, se han desarrollado y evolucionado, junto con una serie de paquetes comerciales de planeación de recursos empresariales (ERP). Estos marcos y paquetes de soporte están haciendo posible que las organizaciones se reinventen en torno a sistemas de sistemas transformadores y centrados en la red.

El uso de un proceso espiral basado en el riesgo con atención temprana a los riesgos y los métodos de arquitectura de sistemas puede evitar muchos de los escollos de desarrollo. Al aplicar la gestión de riesgos están surgiendo las líneas generales de un proceso híbrido basado en un plan / ágil para desarrollar. Para evitar que los desarrollos se desestabilicen debido a grandes cantidades de tráfico de cambios, es importante organizar el desarrollo en incrementos basados en planes en los que los proveedores desarrollen para interactuar con las especificaciones que se mantienen estables al diferir los cambios, para que los sistemas puedan conectarse y reproducirse al final del incremento.

COTS, reutilización y desafíos de integración heredados

Aunque los desarrolladores de software de infraestructura continuarán dedicando la mayor parte de su tiempo a la programación, la mayoría de los desarrolladores de software de aplicaciones pasan cada vez más tiempo evaluando, adaptando e integrando productos COTS. Están evolucionando de forma incontrolable, con un promedio de aproximadamente 10 meses entre las nuevas versiones, y generalmente no cuentan con el respaldo de sus proveedores después de 3 versiones posteriores.

Están surgiendo algunos procesos iniciales de desarrollo de aplicaciones basadas en COTS (CBA) de software. Algunos se basan en elementos de proceso compostables que cubren las principales fuentes de esfuerzo de CBA (evaluación, adaptación e integración de código de pegamento).

Abundancia computacional

La cantidad computacional generará nuevos tipos de plataformas (polvo inteligente, pintura inteligente, materiales inteligentes, nanotecnología, sistemas microelectromecánicos: MEMS) y nuevos tipos de aplicaciones (redes de sensores, materiales adaptables o adaptables, prótesis humanas). Estos presentarán desafíos relacionados con el proceso para especificar sus configuraciones y comportamiento; generar las aplicaciones resultantes; verificar y validar sus capacidades, desempeño y confiabilidad; e integrándolos en sistemas de sistemas aún más complejos.

Sin embargo, además de los nuevos desafíos, la abundancia computacional permitirá enfoques nuevos y más potentes relacionados con el proceso. Permitirá un nuevo y más potente software de autocontrol y computación a través de coprocesadores en chip para verificación de afirmaciones, análisis de tendencias, detección de intrusiones o verificación de código de prueba. Permitirá mayores niveles de abstracción, como la programación orientada a patrones, la programación orientada a múltiples aspectos, el ensamblaje de componentes

visuales orientado al dominio y la programación, por ejemplo, con comentarios de expertos sobre partes faltantes. Permitirá soluciones más simples de fuerza bruta, como la evaluación exhaustiva de casos versus la lógica compleja.

Después de analizar el contexto de las formas más comunes de organizar una fábrica de software, podemos notar que influye el tamaño de la organización y el número de proyectos a atender. Observando que entre más pequeño es el equipo, los integrantes deben participar en todas las etapas del desarrollo, por el contrario, si hay suficientes personas para formar múltiples departamentos centrados en habilidades de software particulares o áreas del ciclo de vida.

En cuanto a las principales oportunidades en el desarrollo de software encontramos algunas consecuencias de separar o no ciertas actividades del ciclo de vida de los sistemas y finalmente que las características de confiabilidad e interoperabilidad serán críticas en el futuro del software.

3. Antecedentes teóricos conceptuales

Marco teórico

SSM

Los problemas duros son problemas caracterizados por el hecho de que están bien definidos. Se asume, en ellos, que hay una solución definida y que se pueden definir metas numéricas específicas a ser logradas. Esencialmente, con un problema duro se puede definir qué tipo de resultado se logrará antes de poner en ejecución la solución. Los " QUÉ " y " los CÓMO " de un problema duro pueden estar determinados previamente en la metodología.

Los problemas suaves, por otra parte, son difíciles de definir. Tienen una componente social y política grande. Cuando pensamos en problemas suaves, no pensemos en problemas sino en situaciones problema. Sabemos que las cosas no están trabajando de la manera en que lo deseamos y queremos averiguar porqué y vemos si hay alguna cosa que podamos hacer para aliviar la situación. Una situación clásica de esto, es que tal vez no sea un " problema " sino una "oportunidad", como es el caso de un proyecto a planear.

La metodología de sistemas suaves fue desarrollada por Peter Checkland para el propósito expreso de ocuparse de problemas de este tipo. Él estuvo en la industria por años trabajando con metodologías de sistemas duros. Él vio cómo éstas eran inadecuadas al ocuparse de problemas complejos que tenían un componente social grande; así en los años 60, él ingresó a la Universidad de Lancaster, localizada en el Reino Unido, en una tentativa de investigar esta área y de ocuparse de estos problemas SUAVES. Su "metodología de sistemas suaves" ["Soft Systems Methodology"] fue creada con base en la investigación en un gran número de proyectos de la industria y su aplicación y refinamiento se concluyeron

años después. La metodología, que es muy agradable cómo lo sabemos hoy, fue publicada en 1981, cuando Checkland vivía de la universidad y tenía pensado perseguir una carrera como profesor e investigador.

SSM se divide en siete etapas distintas. Éstas son;

1. El encontrar hechos de la situación problema. Ésta es una investigación básicamente en el área del problema. ¿Quiénes son los jugadores claves? ¿Cómo trabaja el proceso ahora? etc.
2. Expresar la situación problema con diagramas de Visiones Enriquecidas. En cualquier tipo de diagrama, más conocimiento se puede comunicar visualmente. Un dibujo vale más que 1000 palabras.
3. Seleccionar una visión de la situación y producir una definición raíz. Puede que existan perspectivas diferentes al mirar la situación problema.
4. Modelos conceptuales construidos de lo que hace, las necesidades del sistema para cada una de las definiciones raíz. Usted tiene básico " los qué" de las definiciones de la raíz. Se definen "los cómo".
5. Comparación de los modelos conceptuales con el mundo verdadero. Compare los resultados de los pasos 4 y 2 para ver donde hay diferencias y similitudes.
6. Identifique los cambios factibles y deseables. Hay las maneras de mejorar la situación.
7. Recomendaciones para tomar la acción que mejore la situación problema. Cómo usted pondría práctica los cambios del paso 6. [4]

Evaluación

Actividad programada de reflexión, basada en procedimientos sistemáticos de recolección, análisis e interpretación de información, con la finalidad de emitir juicios valorativos fundamentados y comunicables sobre las actividades, resultados e impactos. Y formular recomendaciones para tomar decisiones que permitan ajustar la acción presente y mejorar la acción futura.

Los tipos más difundidos suelen diferenciarse de acuerdo con los siguientes criterios:

- según el momento en que se realiza la evaluación;
- según quienes realizan, son responsables o intervienen en la evaluación;
- según los propósitos y la índole de las decisiones que se espera tomar en el futuro en función de los resultados de la evaluación;
- según los aspectos a evaluar del programa o proyecto.

Una distinción clásica proveniente de la tradición económica y trasladada a las ciencias sociales es la que suele efectuarse entre la evaluación ex ante y evaluación ex post.

Como su nombre lo indica, la evaluación ex ante (a veces denominada también “de predecisión”, “de factibilidad” o “de pertinencia”) se emprende antes de iniciar un programa o proyecto para tomar una decisión relativa a si debe implementarse o no.

La evaluación ex post es la que se realiza una vez concluida la ejecución del proyecto. Aquí también se utilizan varios términos, “de fin de proyecto”, “a posteriori”, “terminal”. De todos modos, en general se concentra en los resultados obtenidos para evaluar en que medida se alcanzaron los objetivos previstos.[5]

CMM

Modelo de Capacidad y Madurez o CMM (Capability Maturity Model), es un modelo de evaluación de los procesos de una organización.

Fue desarrollado inicialmente para los procesos relativos al software por la Universidad Carnegie-Mellon para el SEI (Software Engineering Institute).

El SEI es un centro de investigación y desarrollo patrocinado por el Departamento de Defensa de los Estados Unidos de América y gestionado por la Universidad Carnegie-Mellon. "CMM" es una marca registrada del SEI.

Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area). Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser:

- Definidas en un procedimiento documentado
- Provistas de los medios y formación necesarios
- Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas)
- Medidas
- Verificadas

A su vez estas Áreas de Proceso se agrupan en cinco "niveles de madurez", de modo que una organización que tenga institucionalizadas todas las prácticas incluidas en un nivel y sus inferiores, se considera que ha alcanzado ese nivel de madurez.

Los niveles son:

1 - Inicial. Las organizaciones en este nivel no disponen de un ambiente estable para el desarrollo y mantenimiento de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven minados por falta de planeación. El éxito de los proyectos se basa la mayoría de las veces en el esfuerzo personal,

aunque a menudo se producen fracasos y casi siempre retrasos y sobrecostes. El resultado de los proyectos es impredecible.

2 - Repetible. En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente.

3 - Definido. Además de una buena gestión de proyectos, a este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detallada y un nivel más avanzado de métricas en los procesos. Se implementan técnicas de revisión por pares (peer reviews).

4 - Gestionado. Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.

5 - Optimizado. La organización completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

Figura 1 Niveles del modelo CMM

Así es como el modelo CMM establece una medida del progreso, conforme al avance en niveles de madurez. Cada nivel a su vez cuenta con un número de áreas de proceso que deben lograrse. El alcanzar estas áreas o estadios se detecta mediante la satisfacción o insatisfacción de varias metas claras y cuantificables. Con la excepción del primer nivel, cada uno de los restantes Niveles de Madurez está compuesto por un cierto número de Áreas Claves de Proceso, conocidas a través de la documentación del CMM por su sigla inglesa: KPA.

Cada KPA identifica un conjunto de actividades y prácticas interrelacionadas, las cuales cuando son realizadas en forma colectiva permiten alcanzar las metas fundamentales del proceso. Las KPAs pueden clasificarse en 3 tipos de proceso: Gestión, Organizacional e Ingeniería.

Las prácticas que deben ser realizadas por cada Área Clave de Proceso están organizadas en 5 Características Comunes, las cuales constituyen propiedades que indican si la implementación y la institucionalización de un proceso clave es efectivo, repetible y duradero.

Estas 5 características son: i) Compromiso de la realización, ii) La capacidad de realización, iii) Las actividades realizadas, iv) Las mediciones y el análisis, v) La verificación de la implementación.[6]

CMMI

CMMI es un modelo para la mejora de procesos que proporciona a las organizaciones los elementos esenciales para procesos eficaces. Las mejores prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay dos áreas de interés cubiertas por los modelos de CMMI: Desarrollo y Adquisición. La versión actual de CMMI es la versión 1.2. Hay dos modelos de la versión 1.2 disponible:

- CMMI para el Desarrollo (DEV-CMMI), Versión 1.2 fue liberado en agosto de 2006. En él se tratan procesos de desarrollo de productos y servicios.
- CMMI para la adquisición (ACQ-CMMI), Versión 1.2 fue liberado en noviembre de 2007. En él se tratan la gestión de la cadena de suministro, adquisición y contratación externa en los procesos del gobierno y la industria.

Independientemente del modelo que opta una organización, las prácticas CMMI deben adaptarse a cada organización en función de sus objetivos de negocio.[7]

Nivel	Focus	Área clave de proceso	Resultado
5 Optimizar	Mejora continua de los procesos	Innovación en organización y despliegue Análisis Causal y Resolución	Máxima calidad y menor riesgo
4 Cuantitativamente gestionado	Cuantitativamente gestionado	Rendimiento del proceso organizacional Gestión de Proyectos cuantitativos	Mayor calidad y menor riesgo
3 Definido	Estandarización de procesos	Desarrollo de requisitos Solución técnica Integración de Productos Verificación Validación Enfoque del proceso organizacional Definición del proceso organizacional Capacitación en materia de organización Gestión integrada de proyectos (IPPD extras) Gestión de Riesgos Análisis de decisión y resolución Equipos Integrados (IPPD) Org. Medio ambiente para la Integración (IPPD) Gestión de Proveedores Integrada (SS solamente)	Calidad media / Riesgo medio
2 Gestionado	Básica para la Gestión de Proyectos	Administración de requisitos Planeación del proyecto Programa de Monitoreo y control Gestión Acuerdo de Proveedor Medición y análisis Proceso y garantía de calidad del producto Gestión de la configuración	Baja calidad / Alto Riesgo
1 Inicial	Proceso es informal y Adhoc		Menor calidad / Alto Riesgo

Tabla 1 Detalle de niveles del modelo CMMI

A partir de los antecedentes teóricos, se puede observar lo valioso que es contar con directrices y estándares recientes y globales, como el Modelo de Madurez de Capacidad Integrada (CMMI) para evaluar un problema suave (como lo es el presente estudio) y aplicar las etapas de una metodología de sistemas suaves para abordarlo.

4. Construcción y modelado de los sistemas comparables

Objeto de estudio físico: Fábrica de software del sector bancario

Modelo conceptual: Proceso de desarrollo de software

La fábrica de software es un área dentro de la Dirección de Sistemas que se encarga de desarrollar software para usuarios internos, proporciona soporte a aquellos servicios desarrollados e implementa software de terceros que cumplen con necesidades específicas.

Definición raíz	
C	Áreas internas de la organización del banco.
A	Ingenieros en computación
T	Automatización de tareas
W	Usuarios que buscan reducir tiempos y errores en su operación
O	Gerente del área
E	Auditorías a los productos entregados

Tabla 2 Definición raíz del objeto de estudio

Ubicación del objeto de estudio

Temporal:

Es de reciente creación, surge de la necesidad de brindar soluciones tecnológicas a la organización, con el objetivo de automatizar tareas repetitivas. Se ha ido consolidando e incrementando su tamaño debido a la demanda de las áreas por contar con herramientas tecnológicas que respalden sus procesos.

Espacial:

La fábrica de software se encuentra en la Ciudad de México. Su ámbito de influencia cubre todo el territorio nacional. El área de desarrollo de sistemas es un área interna que brinda servicios de desarrollo de software a todas las áreas del banco.

Sectorial:

El objeto de estudio pertenece al Sector Terciario o de Servicios dentro de la Actividad Económica de México, teniendo una cobertura a nivel nacional, debido a que su propósito es proporcionar servicios tecnológicos dentro del Sistema Bancario.

Análisis del entorno

Como principales elementos que influyen en su comportamiento se tiene lo siguiente:

El área no tiene competidores directos dentro de la organización, no obstante, la empresa cuenta con otras entidades tecnológicas encargadas de brindar servicios similares a diferentes usuarios. Con estos homólogos se intercambian procedimientos e información acerca de las nuevas tendencias que surgen en el sector.

Weltanschauung (visión del mundo)

Una fábrica de software se dedica al diseño y desarrollo de una aplicación, solución web o de escritorio, que apoya en los diferentes pasos para diagramarla, crearla, documentarla y dejarla en producción en los servidores del cliente.

La fábrica aplica técnicas de manufactura para el desarrollo de software para reproducir los beneficios de la manufactura tradicional (componentes estandarizados, conocimientos especializados de programación, procesamiento en paralelo y un nivel predecible de calidad).

Una fábrica de software ofrece los conocimientos técnicos, metodologías de desarrollo ágil y de implementación que permitan una rápida adquisición de la herramienta, con el fin de solucionar y apoyar la automatización de procesos administrativos o de información, así como los reportes que se definan. Además, de buscar las tecnologías vanguardistas que permitan un rendimiento de la herramienta amigable y eficaz.[8]

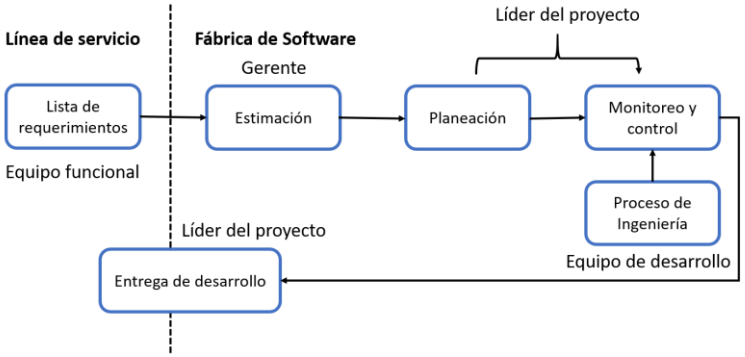

Figura 2 Proceso de administración de proyecto y actores involucrados (Deloitte México 2012)

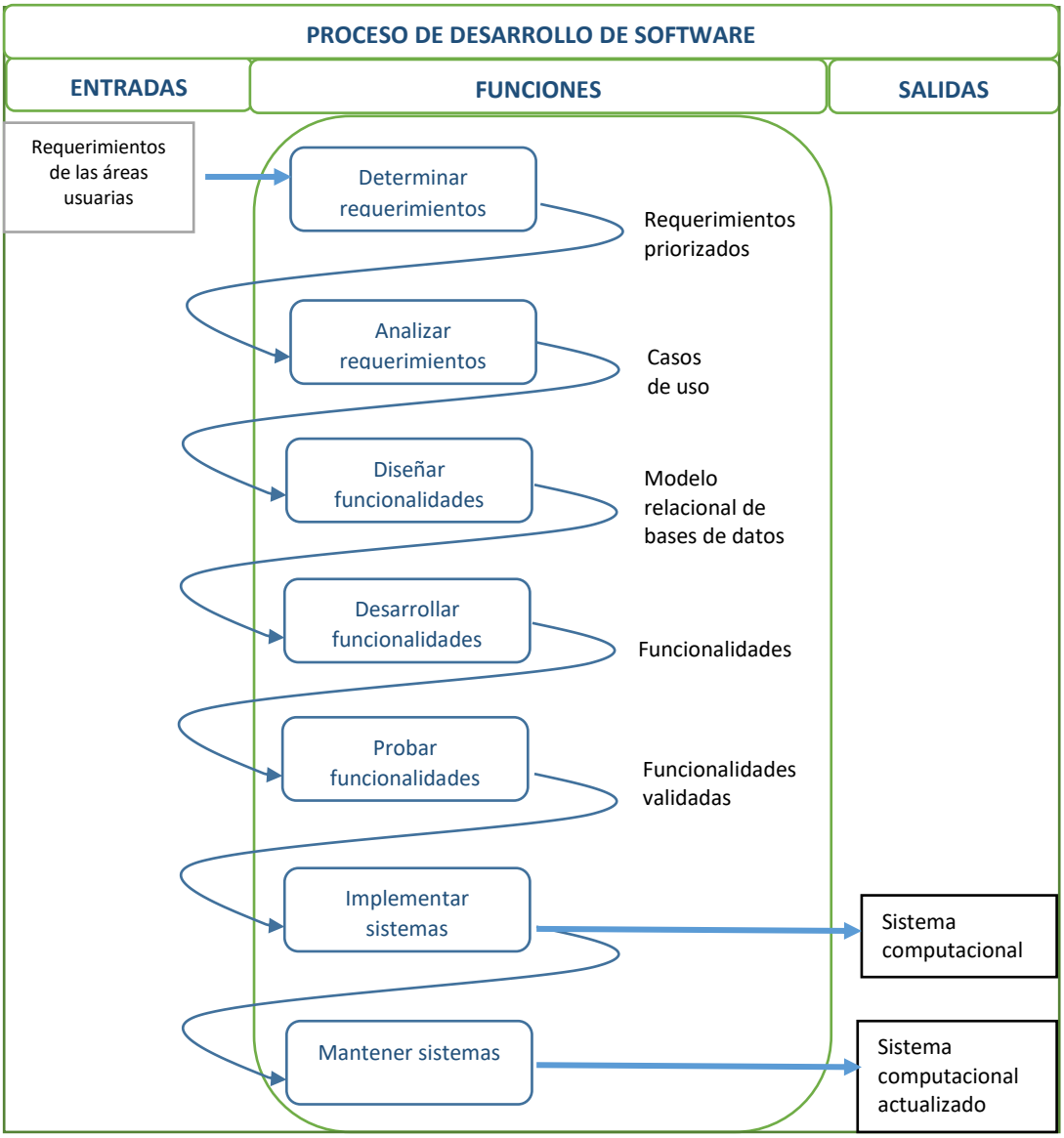

Figura 3 Diagrama de entradas y salidas

Determinar requerimientos

El proyecto de desarrollo de software al igual que cualquier otro proyecto tiene múltiples técnicas de priorización de requerimientos, restricciones presupuestarias y plazos ajustados. Por lo tanto, existe la necesidad de priorizar los requisitos de software ya que es imposible hacer todo a la vez. Por lo tanto, se deben tomar decisiones sobre qué conjunto de requisitos se deben implementar primero y cuáles se pueden retrasar hasta una versión posterior.

Normalmente, los proyectos desean desarrollar software que sea de alta calidad y de gran valor, y la forma más sencilla de desarrollar software de alto valor es implementar primero los requisitos de mayor prioridad. Esto les permite maximizar el ROI de las partes interesadas.

La priorización de requerimientos de software se considera una parte vital del proyecto. Y, por ejemplo, para priorizarlos, podríamos usar los 5 criterios principales de priorización de requerimientos, como el valor que los usuarios le dan a la visión del producto, la urgencia, las limitaciones de tiempo, la técnica de complejidad y las preferencias de las partes interesadas.

Analizar requerimientos

La definición de requisitos o especificación de características que ha de cumplir el software que vamos a desarrollar es la primera etapa. Y probablemente sea la más importante. Al fin y al cabo, lo que sea o no sea el producto final depende de decisiones tomadas en esta etapa. Se trata fundamentalmente de estudiar las necesidades y preferencias del usuario. Es también muy importante dejar clara constancia de las decisiones tomadas en esta etapa, para ser tenidos en cuenta posteriormente. Por ello, la documentación producida en esta fase debe ser concreta y estar siempre disponible durante el resto del proceso.

Diseñar funcionalidades

Una vez planteada la especificación del programa, hay que analizar desde un punto de vista técnico las posibles soluciones. Entre ellas, se elegirá la que se considere más adecuada. A partir de ese momento, se decidirá la estructura general del programa (subdivisión en partes y relaciones entre ellas). Para cada una de las partes se seguirá recursivamente un proceso similar, hasta que tengamos totalmente definido el programa y estemos listos para pasar a la fase de codificación.

En el análisis de cada una de las partes nos encontraremos normalmente con que hay varias soluciones posibles (por ejemplo, varios algoritmos para realizar la misma tarea). La elección de una de ellas suele realizarse de una forma más o menos intuitiva: no hay metodologías efectivas que nos ayuden en esta decisión.

Desarrollar funcionalidades

En un proyecto grande ésta es la etapa más sencilla (en contra de lo que suele suponer cualquier persona que comienza a aprender un lenguaje de

programación). Si el diseño es adecuado y suficientemente detallado la codificación de cada módulo es algo casi automático.

Una de las principales decisiones a tomar en esta fase es la del lenguaje a emplear, aunque a veces en el diseño ya está de alguna forma implícito. Desde hace tiempo la tendencia es a utilizar lenguajes de más alto nivel, sobre todo a medida que se dispone de compiladores más eficientes. Esto ayuda a los programadores a pensar más cerca de su propio nivel que del de la máquina, y la productividad suele mejorarse. Como contrapartida este tipo de lenguajes son más difíciles de aprender. Y además hay que tener en cuenta que los programadores suelen ser conservadores y reacios a aprender nuevos lenguajes: prefieren usar los que ya conocen. La existencia, en una organización, de una gran cantidad de programas desarrollados en un determinado lenguaje, hace además muy dura la decisión de cambiar a uno nuevo.

Probar funcionalidades

En esta fase hay que comprobar que las especificaciones se cumplen perfectamente y en todos los casos. En la realidad es prácticamente imposible probar un programa totalmente: por ello siempre suele quedar algún error escondido. Este problema se agrava cuando sobre él se realizan repetidos cambios y correcciones. Si no los gestionamos de una forma adecuada podemos acabar con un conjunto de parches que más que soluciones aportan problemas.[9]

Implementar sistemas

El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.

Entrenamiento y soporte para el software es de suma importancia y algo que muchos desarrolladores de software descuidan. Los usuarios, por naturaleza, se oponen al cambio porque conlleva una cierta inseguridad, es por ello que es fundamental instruir de forma adecuada a los futuros usuarios del software.

Mantener sistemas

El Servicio de mantenimiento de software es una de las actividades en la Ingeniería de Software y es el proceso de mejorar y optimizar el software desplegado (revisión del programa), así como también depurar los errores.

El mantenimiento de software es también una de las fases en el Ciclo de Vida de Desarrollo de Sistemas (SDLC ó System Development Life Cycle), que se aplica al

desarrollo de software. La fase de mantenimiento es la fase que viene después del despliegue (implementación) del software en el campo.

La fase de mantenimiento de software involucra cambios al software en orden de corregir errores y dependencias encontradas durante su uso tanto como la adición de nueva funcionalidad para mejorar la usabilidad y aplicabilidad del software

Tipos de mantenimiento

A continuación, se señalan los tipos servicio de mantenimientos existentes, y entre paréntesis el porcentaje aproximado respecto al total de operaciones de mantenimiento:

- Perfectivo (60%): Mejora del software (rendimiento, flexibilidad, reusabilidad) o implementación de nuevos requisitos. También se conoce como mantenimiento evolutivo.
- Adaptativo (18%): Adaptación del software a cambios en su entorno tecnológico (nuevo hardware, otro sistema de gestión de bases de datos, otro sistema operativo)
- Correctivo (17%): Corrección de fallos detectados durante la explotación.
- Preventivo (5%): Facilitar el mantenimiento futuro del sistema (verificar precondiciones, mejorar legibilidad).[10]

FUNCIONES	ELEMENTOS	ACTORES
Determinar requerimientos	Requerimientos Casos de negocio	Usuarios funcionales Ingenieros en computación con habilidades de negociación
Analizar requerimientos	Entendimiento de la necesidad Procesos de negocio	Ingenieros en computación con experiencia en modelado de sistemas.
Diseñar funcionalidades	Diagramas	Ingenieros en computación con experiencia en tecnologías, seguridad, mejores prácticas.
Desarrollar funcionalidades	Líneas de código	Ingenieros en computación con experiencia en desarrollo de software.
Probar funcionalidades	Casos de prueba Errores	Ingenieros en computación con experiencia en detección de errores.
Implementar sistemas	Servidores	Ingenieros en computación con experiencia en manejo de servidores.
Mantener sistemas	Rapidez en restablecer el servicio	Ingenieros en computación con experiencia en solución de incidentes y migraciones

Tabla 3 Funciones del modelo conceptual

Definición de las alternativas

Alternativa A

Etapa	Recurso
Determinación de requerimientos	Persona 1- System Analyst
Análisis	Persona 1- System Analyst
Diseño	Persona 1- System Analyst
Desarrollo	Persona 1- System Analyst
Pruebas	Persona 1- System Analyst
Implementación	Persona 1- System Analyst
Mantenimiento	Persona 1- System Analyst

Tabla 4 Definición de la alternativa A

Alternativa B

Etapa	Recurso
Determinación de requerimientos	Persona 1- Business Relationship Manager (BRM)
Análisis	Persona 2- Business Analyst (BA)
Diseño	Persona 3- Software Architect
Desarrollo	Persona 4- Software Development
Pruebas	Persona 5- Tester
Implementación	Persona 6- Deployment Technician
Mantenimiento	Persona 7- Support Analyst

Tabla 5 Definición de la alternativa B

Qué es y qué hace cada alternativa

Alternativa A

Alternativa con la que actualmente cuenta la organización, está integrada por profesionales de software, generalmente cuentan con Licenciaturas en Ingeniería de sistemas o afines y experiencia en desarrollo de software. Él área se encarga de automatizar procesos para las actividades de recursos humanos y se encuentra dividida en equipos de trabajo de acuerdo al negocio que atienden, por ejemplo, se encuentra el equipo encargado de atender al área de nómina, el equipo encargado del plan de salud, otro para pensiones y uno más para becas.

Un mismo equipo es encargado de llevar a cabo todo el ciclo de desarrollo de los sistemas de su negocio, es decir, en primera instancia, el equipo de nómina junto con el área usuaria realiza la determinación de requerimientos considerando

prioridades de atención, después analiza cada uno de estos requerimientos para despejar dudas con los usuarios y garantizar que lo que se solicitó es lo mismo que se ha entendido y que posteriormente se desarrollará.

Continúan con la etapa de diseño que va más orientada a aspectos técnicos de programación, definiendo las herramientas necesarias, la estructura de base de datos y arquitectura del sistema. Enseguida se hace la codificación de cada una de las funcionalidades que integrará el sistema, se procede a probar el sistema con ayuda de los usuarios, se instala en los servidores productivos para que entre en operación y finalmente se le da mantenimiento en el momento que lo requiere.

Además de desarrollar los proyectos nuevos, cada uno de los equipos tiene sistemas que ya se encuentran operando y, por tanto, también se encarga de darles mantenimiento si llega ocurrir algún incidente o actualizar su tecnología en caso de migraciones programadas.

El siguiente esquema representa la organización del área:

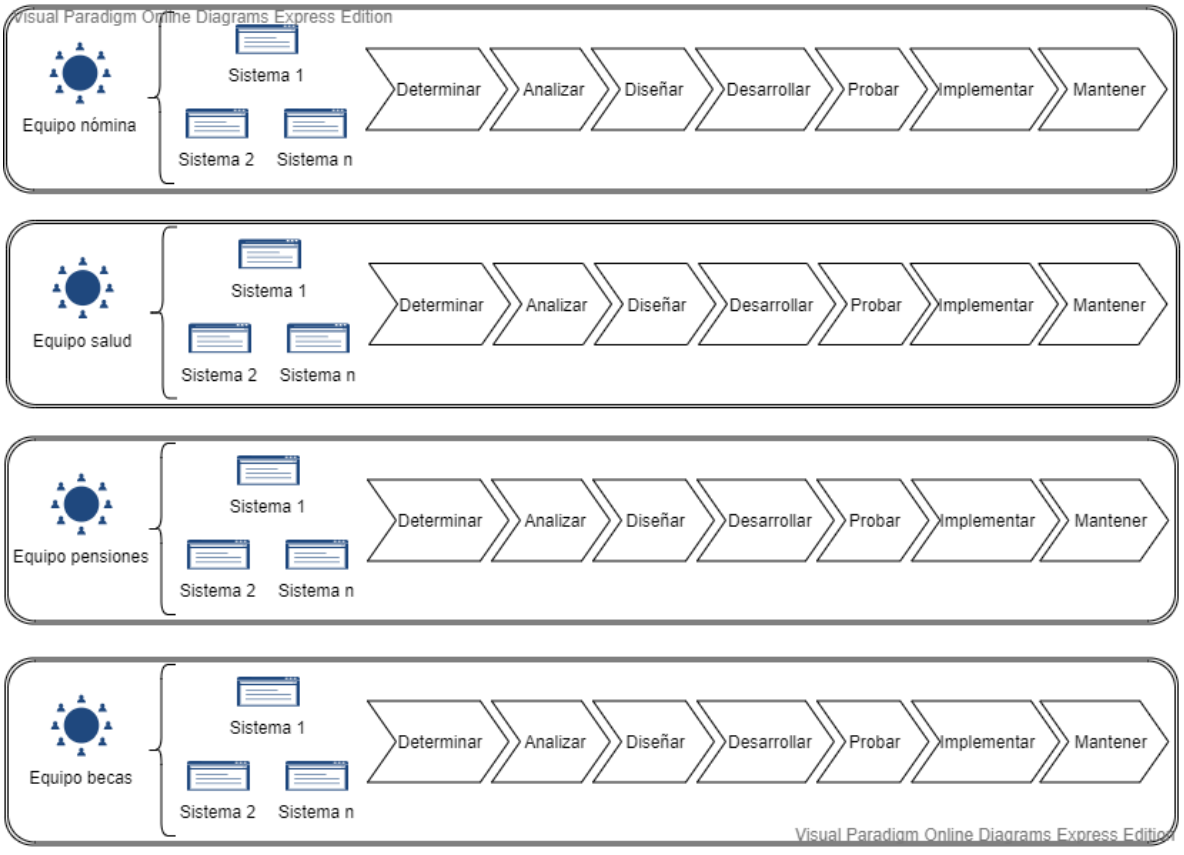

Figura 4 Descripción de la alternativa A

Tomando las cinco funciones más representativas y desglosándolas:

- Determinar requerimientos
  - a. Investigar impacto
    - i. Estimar si hay requerimientos de ley
    - ii. Estimar errores de seguir haciéndolo a mano
  - b. Estimar beneficio
    - i. Estimar horas hombre que se ahorrarían
  - c. Asignar prioridad
    - i. Consultar con las áreas involucradas
- Analizar requerimientos
  - a. Realizar casos de uso
    - i. Definir actores
    - ii. Definir trayectorias de los flujos
  - b. Realizar prototipos
    - i. Definir pantallas
    - ii. Definir controles
- Desarrollar funcionalidades
  - a. Codificar flujo principal
    - i. Crear métodos
    - ii. Crear conexiones a bases de datos
  - b. Implementar autenticación
    - i. Implementar mecanismos de seguridad
- Probar funcionalidades
  - a. Instalar en ambiente de pruebas
    - i. Crear versión de pruebas
    - ii. Generar casos de pruebas
    - iii. Generar datos para los escenarios propuestos
  - b. Corregir errores
    - i. Recrear el error
    - ii. Corregir errores
  - c. Coordinar sesiones con los usuarios
    - i. Capacitar usuarios
- Mantener sistemas
  - a. Garantizar la disponibilidad
    - i. Atender incidentes
  - b. Actualizar infraestructura
    - i. Migrar a tecnología reciente

Alternativa B

Esta alternativa es completamente nueva para el área, sin embargo, algunos corporativos grandes la incluyen en su organización. Consiste en contar con un equipo dedicado y especializado para cada una de las etapas del desarrollo de software. Los equipos de trabajo están integrados generalmente con personas

relacionas al área de tecnologías de la información, pero con experiencia en la etapa en la que se encuentra. Poniendo como ejemplo al equipo encargado de determinar los requerimientos que serán procesados, cuenta con especialistas en casos de negocio, en evaluación y rentabilidad de los proyectos. Mientras que el equipo desarrollador es especialista en mejores prácticas de programación, eficiencia en código y conocimiento de estándares.

Aquí lo que cada equipo debe aprender son las reglas del negocio del sistema con el que estarán trabajando y además se da el caso de estar trabajando en varios proyectos simultáneamente, pero en la misma etapa.

Los equipos trabajan de manera encadenada, es decir, cuando el equipo de determinación de requerimientos terminó su trabajo para el sistema1, entonces pasa la documentación necesaria al equipo de análisis para que pueda continuar con su trabajo y así sucesivamente.

En el siguiente diagrama se representa la estructura de esta alternativa:

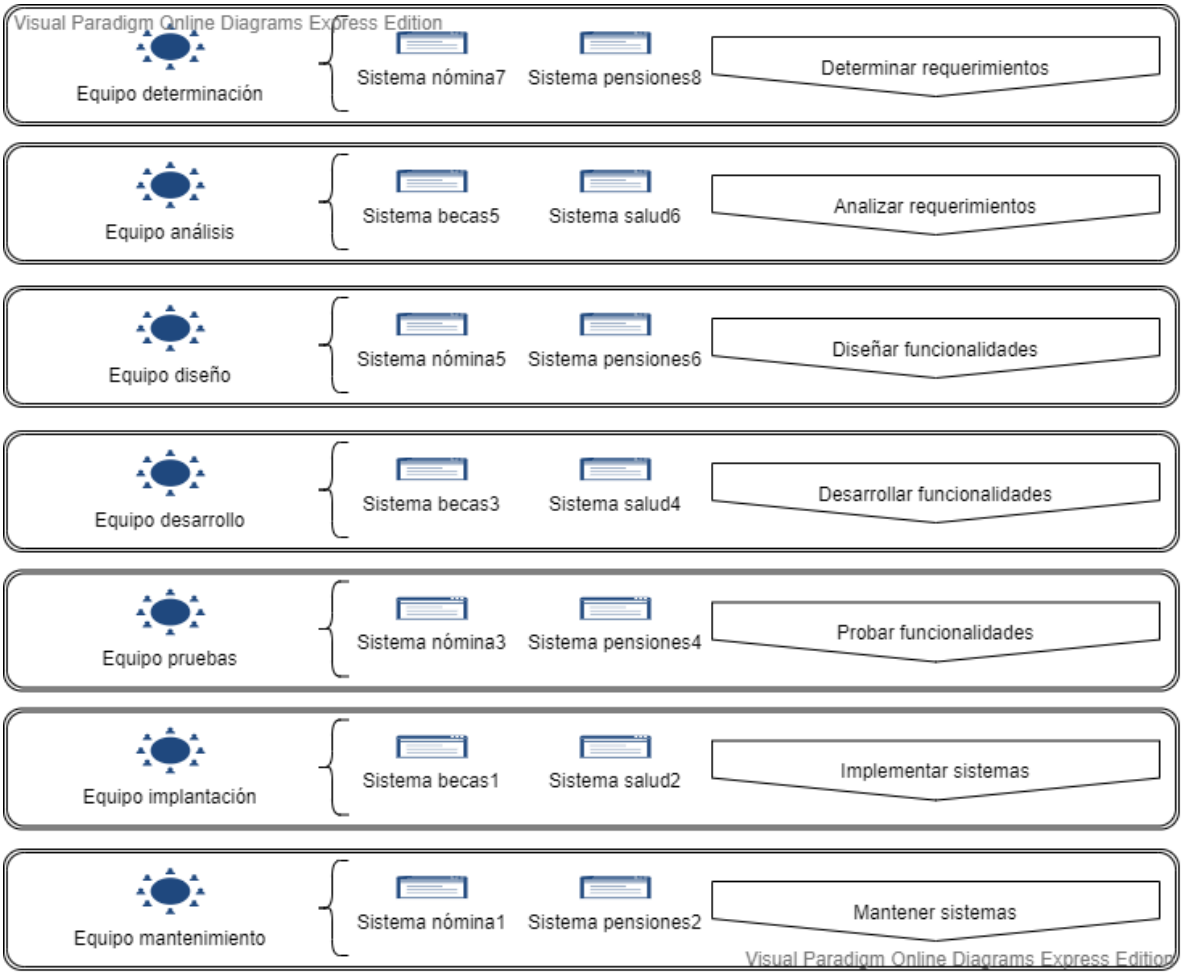

Figura 5 Descripción de la alternativa B

Tomando las cinco funciones más representativas y desglosándolas:

Cada etapa del desarrollo tendrá actividades más especializadas que la alternativa A, debido a que los equipos están dedicados a la etapa en cuestión.

Determinar requerimientos

a. Crear casos de negocio
    i. Calcular indicadores financieros
b. Investigar impacto
    i. Estimar si hay requerimientos de ley
    ii. Estimar errores de seguir haciéndolo a mano
c. Estimar beneficio
    i. Estimar horas hombre que se ahorrarían
d. Asignar prioridad
    i. Consultar con las áreas involucradas

Analizar requerimientos

e. Realizar casos de uso
    i. Definir actores
    ii. Definir trayectorias de los flujos
f. Realizar prototipos
    i. Definir pantallas
    ii. Definir controles
g. Proponer mejoras
    i. Proponer nuevos controles

Desarrollar funcionalidades

h. Codificar flujo principal
    i. Crear métodos
    ii. Crear conexiones a bases de datos
i. Implementar autenticación
    i. Implementar mecanismos de seguridad
j. Implementación de mejores prácticas
    i. Implementación de patrones de diseño
    ii. Implementación de estándares de desarrollo

Probar funcionalidades

k. Instalar en ambiente de pruebas
    i. Crear versión de pruebas
    ii. Generar casos de pruebas
    iii. Generar datos para los escenarios propuestos
l. Corregir errores

i. Recrear el error
ii. Corregir errores

m. Coordinar sesiones con los usuarios
    i. Capacitar usuarios
n. Pruebas automatizadas
    i. Pruebas de caja negra
    ii. Pruebas de caja blanca
    iii. Pruebas de penetración
    iv. Pruebas que permitan encontrar vulnerabilidades de seguridad

Mantener sistemas

o. Garantizar la disponibilidad
    i. Atender incidentes
p. Actualizar infraestructura
    i. Migrar a tecnología reciente
q. Garantizar acuerdos de niveles de servicio
    i. Seguimiento de incidentes
    ii. Clasificación de errores
    iii. Creación de bitácora de errores conocidos
    iv. Clasificación de problemas

5. Evaluación de alternativa A

Para evaluar cada alternativa de este estudio se utilizó el modelo CMMI ya que es el referente mundial en la evaluación de la calidad y la aplicación de las mejores prácticas en desarrollo de software.

Nivel de madurez

Con el objetivo de conocer el nivel de madurez que cuenta el área de informática se elaboró y aplicó un cuestionario de treinta preguntas.

Para la elaboración del cuestionario se seleccionaron nueve de las áreas clave que considera el modelo CMMI por cada nivel de madurez y se convirtieron en nuestros criterios de evaluación. Para cada uno de estos criterios se clasificaron tres preguntas en promedio, quedando de la siguiente manera[11]:

Criterio	Preguntas
Innovación en organización y despliegue	3
Análisis causal y resolución	3
Rendimiento del proceso organizacional	5
Gestión de proyectos cuantitativos	3
Verificación (¿el software cumple con sus requisitos?)	3
Validación (¿hace el software lo que el usuario realmente requiere?)	4
Gestión de riesgos	3
Planeación del proyecto	3
Proceso y garantía de calidad del producto	3
	30

Tabla 6 Desglose de preguntas del cuestionario

Cada pregunta debía ser contestada en un rango de 1 (Totalmente en desacuerdo) a 5 (Totalmente de acuerdo). Además, el cuestionario se capturó en una herramienta en línea que permitiera su fácil distribución y aplicación.

Considerando que el área de informática cuenta un total de 36 personas, el tamaño de la muestra resultó en n=24 (Anexo 1 Cálculo del tamaño de la muestra) tomando en cuenta los siguientes datos:

N	tamaño de la población	=36.0
Z	nivel de confianza	=1.96
p	probabilidad de éxito	=0.05
q	probabilidad de fracaso	=0.95
d	precisión	=0.05

Para seleccionar a las personas se eligieron a 24 de ellas tomando en cuenta que fueran de distintos equipos de trabajo, teniendo entonces la opinión de un 66.67% de la población.

Durante la aplicación del cuestionario, se notaron comentarios diversos, algunos de los cuales se enlistan a continuación:

- "Pues no estamos tan mal, ¿no?"
- "Oye es que, si tenemos tal herramienta para la documentación de errores, ¡¡¡pero no la ocupamos!!!".
- “Yo creo que, si implementamos tal cosa, podemos mejorar nuestra calificación de la categoría de validación”

Es decir, se pudo notar el entusiasmo de los participantes por expresar sus ideas para mejorar ciertos aspectos.

Resultados

Se promediaron los resultados obtenidos para cada categoría y se ordenaron de menor a mayor, teniendo a la categoría “Verificación” como la más baja evaluada y a “Gestión de Riesgos” como la categoría mejor evaluada.

El promedio total fue de 3.24 entrando en el nivel 3 del modelo CMMI considerado como Calidad media / Riesgo medio. A continuación, se muestra el detalle de las categorías:

Categoría	Promedio de la categoría
Verificación (¿el software cumple con sus requerimientos?)	2.86
Gestión de proyectos cuantitativos	2.94
Validación (¿hace el software lo que el usuario realmente requiere?)	2.98
Innovación en organización y despliegue	3
Proceso y garantía de calidad del producto	3.11
Rendimiento del proceso organizacional	3.23
Análisis causal y resolución	3.25
Planeación del proyecto	3.44
Gestión de riesgos	4.33

Tabla 7 Promedio por categorías

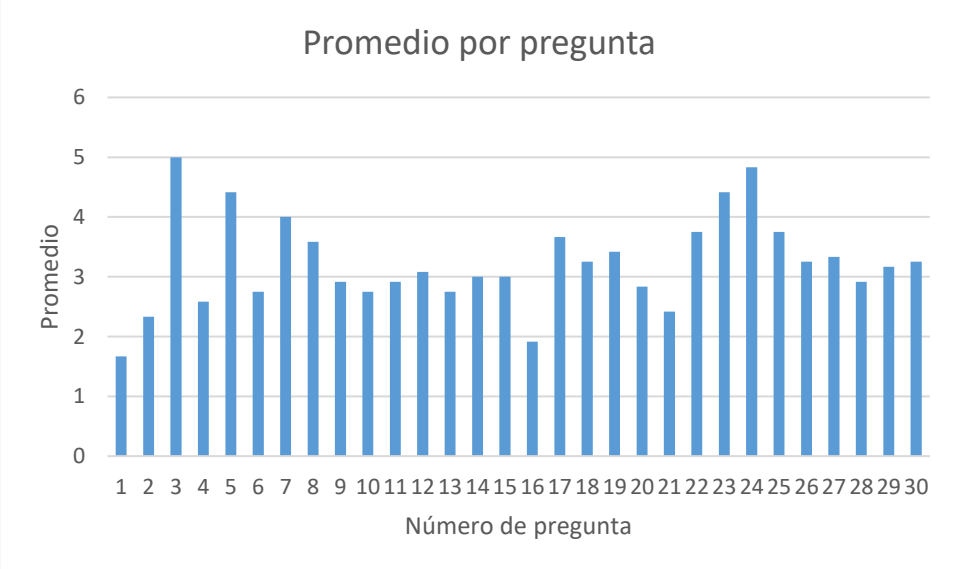

Figura 6 Análisis del promedio por pregunta

Adicionalmente, se realizó un análisis de la desviación estándar de cada pregunta, obteniendo que el valor más alto es de 1.44 para la pregunta 17 lo que nos lleva a resumir que la opinión de los encuestados es muy cercana ya que incluso para una pregunta la opinión fue uniforme (pregunta número 3).

A continuación, se muestran los resultados de este análisis:

Categoría	Desviación estándar promedio por categoría
Innovación en organización y despliegue	0.52
Gestión de riesgos	0.93
Análisis causal y resolución	1.12
Planeación del proyecto	1.17
Gestión de proyectos cuantitativos	1.17
Rendimiento del proceso organizacional	1.18
Validación (¿hace el software lo que el usuario realmente requiere?)	1.2
Proceso y garantía de calidad del producto	1.25
Verificación (¿el software cumple con sus requerimientos?)	1.28

Tabla 8 Desviación estándar por categorías

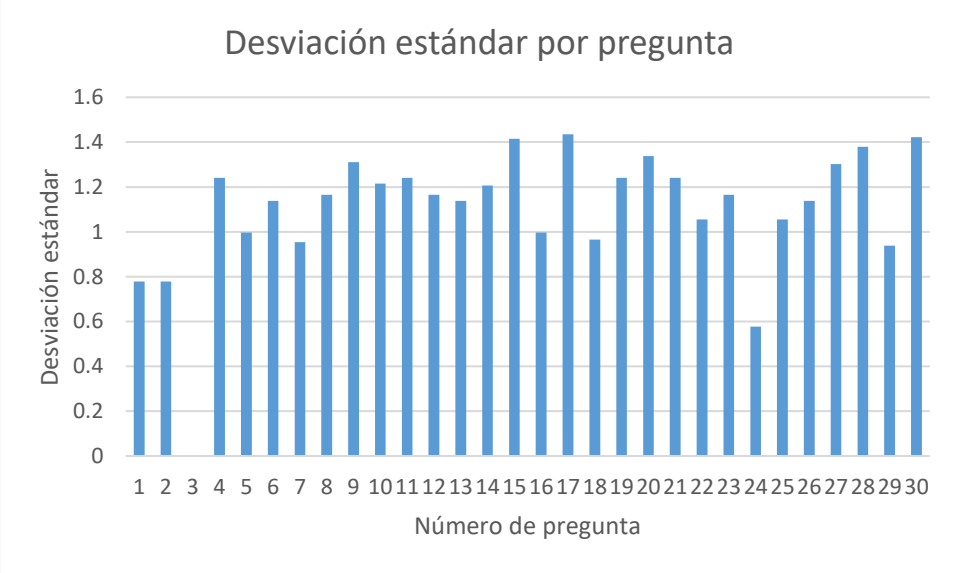

Figura 7 Análisis de desviación estándar por pregunta

Y el detalle de categorías junto sus preguntas se enlistan a continuación:

Categoría	#	Pregunta
Innovación en organización y despliegue	1	¿Existe algún área encargada de lograr reducciones en los tiempos de proceso?
	2	¿Considera adecuada la frecuencia con que se realiza el monitoreo de cada proceso?
	3	¿Es necesaria la investigación para ayudar a las organizaciones a desarrollar y mantener servicios de calidad?
Análisis Causal y Resolución	4	¿Actualmente su área cuenta con estadísticas de los errores y soluciones de cada proceso?
	5	¿Considera que la solución a un problema puede ser aplicada a otros problemas?
	6	¿Documentan las deficiencias para adoptar medidas correctivas o preventivas?
Rendimiento del proceso organizacional	7	¿Conoce los objetivos del área para el cual trabaja?
	8	¿Cuándo realiza sus funciones cuenta con algún procedimiento que le permita alcanzar el objetivo del área?
	9	¿Se recibe capacitación para la correcta aplicación de los lineamientos de los procesos organizacionales?
	10	¿La interacción de área con otros departamentos es eficiente?
	11	¿Percibe adecuado el tiempo de respuesta hacia los clientes?
Gestión de Proyectos cuantitativos	12	¿Se lleva un control estricto de los cambios de un producto, durante su ciclo de vida?
	13	¿Se realiza periódicamente una recolección de la información para analizar el estado del proyecto?
	14	¿Sabe si la gerencia mantiene algún acuerdo o compromiso de productividad con su área?
Verificación (¿El software cumple con sus requisitos?)	15	¿Son frecuentes las inconsistencias entre los requisitos y los planes de un proyecto?
	16	¿Su entidad tiene reuniones planeadas para discutir la calidad del servicio que proporcionan?
	17	¿Se garantiza la integridad de los datos conforme se realizan actualizaciones entre módulos de operación?
Validación (¿Hace el software lo que el usuario realmente requiere?)	18	¿Son efectivos los métodos que utiliza la organización para definir las necesidades de un servicio?
	19	¿Participan de manera apropiada las partes interesadas en un proyecto?
	20	¿Considera elevada la satisfacción de los clientes (usuarios de los sistemas)?
	21	¿Cuentan con algún mecanismo que les permita evaluar la satisfacción de los clientes?
Gestión de Riesgos	22	¿Se cuentan con los procedimientos adecuados para garantizar la continuidad del servicio que presta la dependencia en caso de fallar los sistemas de cómputo?
	23	¿Se tienen identificados los servicios críticos a mantener durante y después de una contingencia?
	24	¿Se cuentan con lugares alternos para la prestación de los servicios críticos durante contingencias?
Planeación del proyecto	25	¿La organización diseña planes de trabajo para alcanzar los objetivos de un proyecto al inicio del mismo?
	26	¿Los planes de trabajo, establecidos en su dependencia, se cumplen?
	27	¿Los planes de trabajo son alcanzables de acuerdo a las habilidades de cada integrante del grupo de trabajo?
Proceso y garantía de calidad del producto	28	¿La organización verifica que el producto o servicio, satisface los requerimientos del cliente?
	29	¿Se realizan análisis y actualizaciones a los procesos organizacionales?
	30	¿Son seleccionadas adecuadamente las herramientas tecnológicas para mejorar la calidad productiva de la organización?

Tabla 9 Detalle del cuestionario

Se puede notar que la alternativa A obtuvo menor puntuación en las categorías de verificación y validación del software, debido a que no se cerciora si el software cumple con lo solicitado y si realmente cubre las necesidades de los usuarios, por tanto, se propone un agente evaluador que se detallará más adelante.

6. Evaluación de alternativa B

Debido a que la alternativa B es un diseño al cual se aspira y no se tuvo acceso a un modelo implementado que se pudiera estudiar, se recurrió a la literatura para indagar esta alterativa.

Estudios de caso

Implementación del modelo CMMI para la creación de fábrica de software de la UTSJR[12]

En la región de San Juan del Rio están instaladas el 37.10 % de las empresas del estado de Querétaro.

Dada su actividad empresarial, se hace necesaria la adquisición de servicios tecnológicos para estar a la vanguardia de la innovación en el área de TI.

El cuerpo Académico de TIC, de la Universidad Tecnológica de San Juan del Rio, encuentra las condiciones idóneas para establecer una fábrica de software que ofrezca servicio, innovación, alternativas y solución a las necesidades, requerimientos y problemáticas que enfrentan los distintos sectores de la sociedad dado que no existe en la región una empresa que funcione como fábrica de Software y que además cumpla con la normas y estándares pertinentes de la industria.

Para cumplir con las normas y estándares de calidad en el desarrollo de software la Universidad Tecnológica de San Juan del Río planea aplicar un conjunto de prácticas de desarrollo de software asociadas al modelo CMMI nivel de madurez 2, para construir los procedimientos, procesos, guías, formatos y herramientas necesarias para soportar el modelo.

- Comparación con la alternativa A actualmente implementada en el Banco:

Se puede notar que la forma en que está organizado actualmente el área de informática del banco permite alcanzar un nivel de madurez 3 con base en el modelo CMMI, superior al que pretende alcanzar la Universidad Tecnológica de San Juan del Rio. Por tanto, no nos representa un modelo útil a considerar, esto podría deberse a que la fábrica de software que mencionan apenas va a iniciarse, no trata de mejorar aún.

Ayesa alcanza el máximo nivel de CMMI para desarrollo de software internacional[13]

Ayesa, multinacional española especializada en ingeniería, tecnología y consultoría, ha logrado el nivel 5 de CMMI, con lo que consigue la máxima categoría para su servicio de software factory.
Esta certificación del organismo estadounidense CMMI Institute es la más prestigiosa a nivel internacional para empresas de desarrollo de aplicaciones y programas informáticos. Sólo un reducido número de empresas certificadas a nivel mundial ha alcanzado este nivel, mientras que en España la cifra se limita a una docena.
El nivel 5 de CMMI (Capability Maturity Model Integration) representa un sello de calidad y excelencia para los proyectos de Ayesa, en términos de ejecución, procesos, fiabilidad y seguridad. Además, implica una mejora continuada del proceso de desarrollo y optimización de los objetivos de la compañía.
Ayesa desarrolla software para múltiples sectores y tiene actividad en varios países de Europa y Latinoamérica. La certificación se extiende a toda la actividad de software factory por un periodo de tres años y permitirá a la compañía acceder a las licitaciones internacionales más exigentes.

- Comparación con la alternativa A actualmente implementada en el Banco:

Sin lugar a dudas la forma en la que Ayesa crea software debe ser mediante un proceso muy riguroso para que le permitiera alcanzar el máximo nivel en el modelo de madurez. Representa un modelo a seguir para verificar las buenas prácticas que se pueden adoptar, sin embargo, no expresan mayor detalle de la manera en que desarrollan el software, no podemos saber si se trata de la alternativa A con algunas modificaciones o de la alternativa B.

Stacks alcanza el nivel 2 de CMMI en su Barcelona Health Software Factory[14]

Recientemente, Stacks ha sido evaluada satisfactoriamente en el nivel 2 del Capability Maturity Model Integration (CMMI) del CMMI Institute. La evaluación fue llevada a cabo por Victoria Ines Lázaro Lamoratta, de Sopra Group Informatica, S.A., como último paso a un proyecto de transformación de procesos iniciado a mediados del 2013.

El alcance de la evaluación se circunscribió a Barcelona Health Software Factory, que conforma a los equipos de desarrollo dedicados a los productos OMI y MLM, vertientes nacional e internacional respectivamente de la solución cloud para el sector sanitario que ofrece Stacks.

El proyecto se inició con una evaluación inicial del estado de los procesos respecto a los requisitos del modelo CMMI, ello permitió identificar las

oportunidades de mejora y definir las correspondientes acciones de mejora a llevar cabo. Con base en a las recomendaciones anteriormente elaboradas se pasó a definir el Plan Estratégico de Mejora de Procesos que permitiera alcanzar el objetivo de acreditación establecido.

Se iniciaron entonces las fases de formación e implantación de los procesos conforme al modelo CMMI, culminando con la acreditación formal en el Nivel de Madurez/Capacidad objetivo.

Una evaluación de la madurez a nivel 2 de CMMI confirma que las actividades se desarrollan de forma “gestionada”. A este nivel, los proyectos se desarrollan sobre una base común de procesos planificados y ejecutados de acuerdo con la política establecida. Dichos proyectos son llevados a cabo por personas debidamente formadas y que disponen de los recursos adecuados para la obtención de resultados controlados. Los proyectos desarrollados bajo este modelo involucran a toda la organización y a los stakeholders relevantes de Stacks y en todo momento están monitorizados, controlados y revisados, así como evaluados en cuanto a su adherencia respecto a los procesos descritos.

- Comparación con la alternativa A actualmente implementada en el Banco:

Al igual que en el primer caso, Stacks alcanza un nivel de madurez inferior al que actualmente cuenta el banco, por tanto, no nos sirve de modelo para mejorar. Lo que sí es importante señalar es que hace énfasis de las ventajas de un nivel de madurez nivel dos: “los proyectos se desarrollan sobre una base común de procesos planificados y ejecutados de acuerdo con la política establecida”, ventaja con la que ya dispone el actual proceso del banco.

El caso 2 (Ayesa), es el que alcanza el nivel de madurez más alto de los casos estudiados, lo cual nos permite inferir que su forma de organizar su área de software es un modelo a seguir para que el banco pueda obtener un mayor nivel CMMI. Sin embargo, hay un inconveniente, el tamaño de Ayesa es muy superior a el banco ya que tiene actividad en varios países de Europa y Latinoamérica.

Por tanto, pensando a mediano plazo, al banco le convendría continuar con un modelo como el que lleva hasta ahora ya que con un nivel de madurez 3 es competitivo, sin embargo, no se descarta un modelo híbrido que permita mejorar sus puntos débiles.

7. Conclusiones y recomendaciones

Actualmente con la alternativa A, el banco cuenta con un nivel de madurez 3, un nivel aceptable y competitivo. Para alcanzar un nivel más alto a mediano plazo no se recomendaría cambiar drásticamente a un esquema en donde las actividades del ciclo de vida de software las realice un equipo dedicado a cada tarea debido al tamaño de su fábrica de software.

Sin embargo, después de realizar el cuestionario, se encontraron los principales puntos débiles (que se describen en la siguiente tabla), se recomienda optar por un modelo hibrido que permita reforzar estos puntos débiles que sobresalieron en la evaluación (validación, verificación de software y gestión de proyectos cuantitativos) y que efectivamente se reflejan en la problemática.

Oportunidad destacada en la evaluación	Síntomas expresados en la problemática	Resultado en la evaluación	Forma de mejorar
Validación (¿hace el software lo que el usuario realmente requiere?)	Incidentes detienen los procesos del área usuaria.	2.98	Modelo híbrido que integra un equipo dedicado a la etapa de pruebas.
Gestión de proyectos cuantitativos	Los incidentes deben atenderse de manera prioritaria y los analistas asignados a proyectos deben detener sus desarrollos provocando retrasos en los proyectos.	2.94	
Verificación (¿el software cumple con sus requerimientos?)	Cantidad de errores que presentan las aplicaciones en ambiente productivo no es controlada.	2.86	

Tabla 10 Principales oportunidades encontradas en la evaluación

El modelo híbrido propuesto consiste en separar solamente a un equipo dedicado a la etapa de pruebas, lo cual conserva las principales ventajas del sistema actual

y mitiga uno de sus principales problemas, a continuación, se detallan las características de este modelo híbrido:

- Un desarrollador conoce de primera mano los tiempos que toma desarrollar ciertas funcionalidades y por tanto puede hacer mejores estimaciones.
- Los desarrolladores cuestionan directamente a los usuarios, lo que permite el conocimiento del negocio para mejores análisis y diseños, ya que no hay una capa adicional que pueda malinterpretar la información.
- Al integrar un equipo dedicado a probar, se le quita esa responsabilidad al desarrollador ya que no puede ser juez y parte de su trabajo, sumado a ello, un especialista en encontrar errores permitiría la entrega de software con más calidad. Y también puede comparar si lo solicitado es lo mismo que se está entregando.
- De la mano al punto anterior, si se cuenta con software que contiene el mínimo de errores, su mantenimiento será menor. Por tanto, los desarrolladores no gastaran tanto tiempo en depuraciones.
- Además, se conservan las ideas de que, si la misma persona que programa, es también la que mantiene los sistemas, entonces se esmera en entregar software de calidad desde las primeras versiones, y si se trata de la misma persona que realizó el sistema le tomará menos tiempo su depuración.

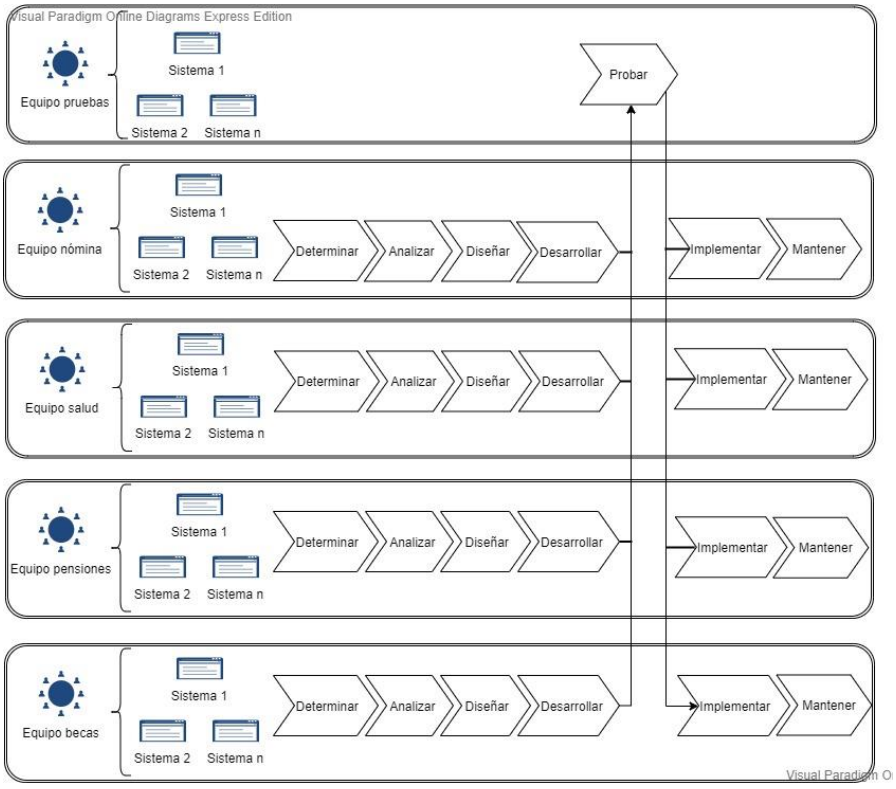

Figura 8 Propuesta de modelo híbrido

8. Lecciones aprendidas

- Cada decisión tomada en las etapas del proceso de desarrollo de software tiene repercusión en sus etapas sucesivas, ya sea positiva o negativa.
- Ninguna de las alternativas presentadas en este estudio (alternativa A y alternativa B) garantiza un nivel 5 en el modelo CMMI.
- El trabajo realizado fue importante porque a primera instancia se creía que mejorar el rendimiento de la fábrica de software se lograría optando por la alternativa B, ya que en varios lugares se usa, pero al analizarla y ver la cantidad de personas que tenemos y el tamaño de los proyectos no nos convenia adoptar este modelo si no, reforzar los puntos débiles y conservar los puntos fuertes (modelo híbrido).
- No existe una forma universal para organizar una fábrica de software ya que eso depende del tamaño de la organización y del número y tamaño de los proyectos a atender. Sin embargo, si se puede optimizar para obtener un mayor nivel de madurez en el modelo CMMI.

9. Trabajo a futuro

Se podría elaborar un modelo de simulación para la alternativa A y para la alternativa hibrida, con el objetivo de estimar que tanto más mejora el rendimiento. Considerando el mismo tamaño de la organización, personas y proyectos se podría realizar una comparación al mismo nivel de granularidad.

10. Anexos

Anexo 1 Cálculo del tamaño de la muestra

Cálculo del tamaño de la muestra conociendo el tamaño de la población[15]

La fórmula para calcular el tamaño de muestra cuando se conoce el tamaño de la población es la siguiente:

$$n = \frac{NZ_a^2pq}{d^2(N-1) + Z_a^2pq}$$

En donde:

N = tamaño de la población
Z = nivel de confianza
p = probabilidad de éxito, o proporción esperada
q = probabilidad de fracaso
d = precisión (Error máximo admisible en términos de proporción)

Concretamente para el presente estudio:

N = 36
Z = 1.96 (confianza 95%)
p = 0.05
q =(1-p) =1-0.05= 0.95
d = 0.05

$$n = \frac{NZ_a^2pq}{d^2(N-1) + Z_a^2pq} = \frac{(36)(1.96)^2(0.05)(0.95)}{(0.05)^2(36-1) + (1.96)^2(0.05)(0.95)} = 24.33$$

Anexo 2 Comportamiento de la muestra

La muestra es una parte representativa de una población donde sus elementos comparten características comunes o similares.

Se utiliza para estudiar a la población de una forma más factible, debido a que se puede contabilizar fácilmente. Cuando se va a realizar algún estudio sobre el comportamiento, propiedades o gustos del total de una población específica, se suelen extraer muestras.

Estos estudios que se realizan a las muestras sirven para crear normas o directrices que permitirán tomar acciones o simplemente conocer más a la población estudiada.

El muestreo es una herramienta de investigación que, al ser utilizada adecuadamente, permite obtener conclusiones específicas y evitar resultados sesgados.

Las principales ventajas de usar las muestras es la reducción de costos, pues disminuye los elementos a estudiar y se puede realizar en menor tiempo. [16]

En la siguiente tabla se puede observar cómo al aumentar la población, el tamaño de la muestra resulta ser un porcentaje menor de la población. Por lo que para el cálculo de la muestra del presente trabajo resultó ser un porcentaje alto de la población total.

N	Z	p	q	d	tamaño de la muestra	% de la población
10	1.96	0.05	0.95	0.05	8.902310514	89.02310514
20	1.96	0.05	0.95	0.05	15.86913417	79.34567085
30	1.96	0.05	0.95	0.05	21.46978539	71.56595131
36	1.96	0.05	0.95	0.05	24.3322962	67.58971168
40	1.96	0.05	0.95	0.05	26.07023459	65.17558648
50	1.96	0.05	0.95	0.05	29.91645244	59.83290488
60	1.96	0.05	0.95	0.05	33.17986763	55.29977938
70	1.96	0.05	0.95	0.05	35.98361579	51.40516542
80	1.96	0.05	0.95	0.05	38.41842643	48.02303303
90	1.96	0.05	0.95	0.05	40.55262534	45.0584726
100	1.96	0.05	0.95	0.05	42.43864774	42.43864774
200	1.96	0.05	0.95	0.05	53.67130605	26.83565302
500	1.96	0.05	0.95	0.05	63.80386804	12.76077361
1000	1.96	0.05	0.95	0.05	68.08866945	6.808866945
10000	1.96	0.05	0.95	0.05	72.46869497	0.72468695

Tabla 11 Comportamiento del tamaño de la muestra en medida que aumenta la población

Anexo 3 Diseño del cuestionario

El cuestionario se plasmó en una herramienta web para su mejor distribución, se puede encontrar en la siguiente liga:

https://docs.google.com/forms/d/1GfXCTShAdj2rZo_wdsCv7l7t5k9XQKiz-krL6rsGCZo/edit?usp=sharing

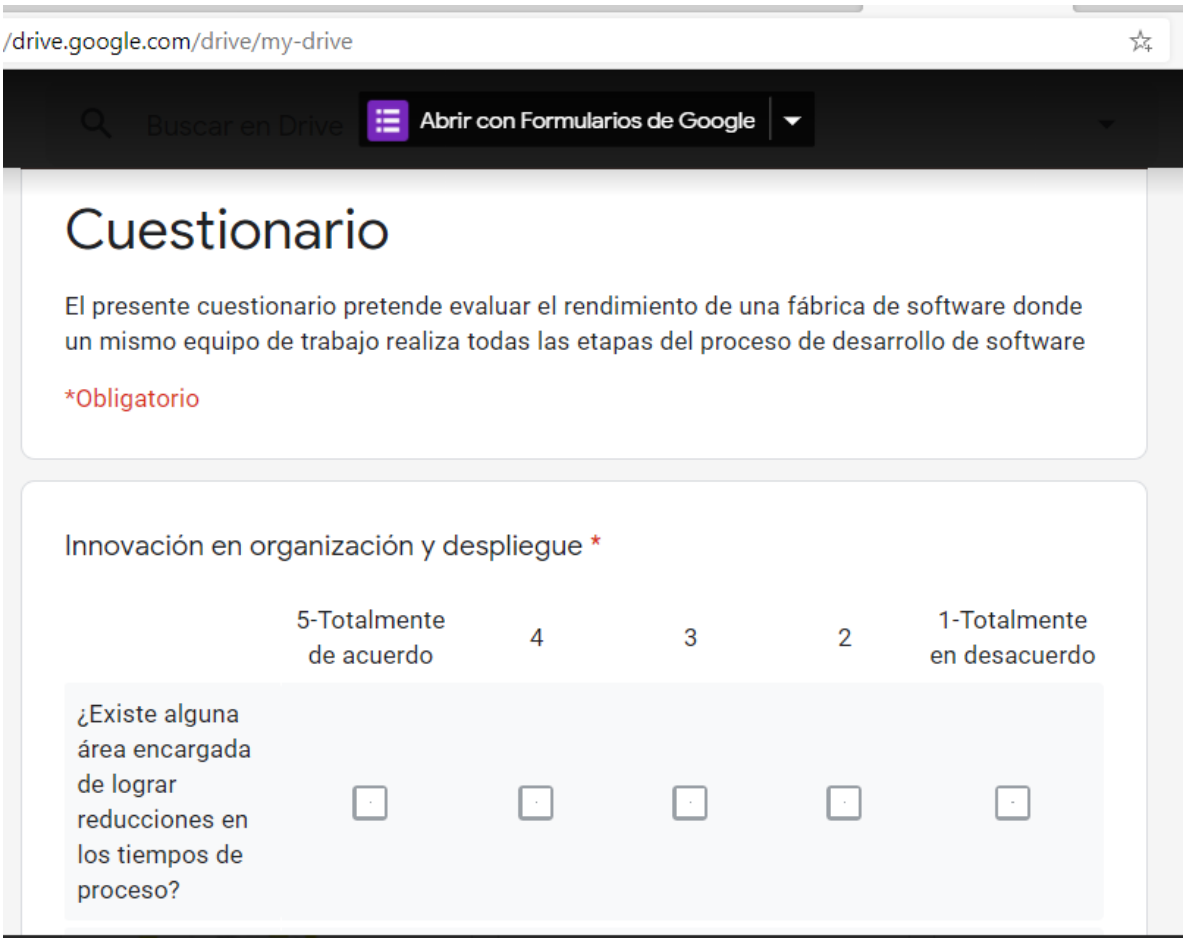

Figura 9 Visualización del cuestionario

Glosario

Ambiente de desarrollo: este es el lugar donde realizamos todos los cambios locos que se nos ocurran, nuevas ideas, o ajustes del cliente. Nada de lo que se realice acá afectará realmente la aplicación de software. OrtizVivas https://ortizvivas.com/blog/ambientes-de-trabajo-para-software/

Ambiente de producción: en este ambiente es donde ocurren todas las transacciones de los usuarios en caliente. Dependiendo de que haga el software puede ser de mucho o poco movimiento, pero al final es la realidad de los usuarios. Los cambios que se han solicitado se recomienda realizarlos en horas de poco o ningún acceso. OrtizVivas https://ortizvivas.com/blog/ambientes-de-trabajo-para-software/

Ambiente de pruebas: normalmente es un lugar que tiene casi los mismos datos del ambiente productivo, pero que se le implementan los cambios realizados en desarrollo. Estos cambios deben haber pasado por las pruebas iniciales y se trasladan a pruebas para ver la integración. En este ambiente es donde el usuario final intenta comprobar que lo que pidió en realidad ocurra. OrtizVivas https://ortizvivas.com/blog/ambientes-de-trabajo-para-software/

Depuración de programas: es el proceso de identificar y corregir errores de programación. En inglés se conoce como debugging, porque se asemeja a la eliminación de bichos (bugs), manera en que se conoce informalmente a los errores de programación. Wikipedia

Error de software: error o simplemente fallo (también conocido por el inglés bug) es un problema en un programa de computador o sistema de software que desencadena un resultado indeseado. Los programas que ayudan a la detección y eliminación de errores de software son denominados depuradores (en inglés, debuggers). Wikipedia

Medida de la calidad de software: descomponiendo atributos, para no tener márgenes de error e interpretación: Atributo de funcionalidad. Atributo de capacidad de respuesta frente a errores externos. Atributo de nivel de seguridad. La calidad no puede existir sin seguridad, un producto sin seguridad seria un producto sin calidad. El observador o usuario final indica que atributos son más o menos importantes en cuanto a seguridad. Wikipedia

Migración informática: es el proceso mediante el cual realizamos una transferencia de datos de unos sistemas de almacenamiento de datos a otros, de unos formatos de datos a otros o entre diferentes sistemas informáticos. Habitualmente, un proyecto de migración de datos se lleva a cabo para reemplazar o actualizar servidores o equipos de almacenamiento, para una consolidación de un sitio web, para llevar a cabo el mantenimiento de un servidor o para reubicar un centro de datos. PowerData https://www.powerdata.es/migracion-de-dato

Parche informático: Un parche es un "paquete" de código utilizado para mejorar un programa, arreglar fallos, introducir nuevas funcionalidades o cualquier otro cambio que se quiera introducir en un programa ya existente. Sánchez, Alberto (22 de agosto de 2019)

Pruebas de caja blanca: (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El ingeniero de pruebas escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados. Wikipedia

Pruebas de caja negra: es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software. Pmoinformatica http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html

Pruebas de penetración: (también llamadas "pen testing") son una práctica para poner a prueba un sistema informático, red o aplicación web para encontrar vulnerabilidades que un atacante podría explotar. https://searchdatacenter.techtarget.com/

Tamaño de un proyecto de software: se realiza de acuerdo a las siguientes estimaciones: estimación de la duración de su proyecto, estimación del esfuerzo que costará realizarlo y estimación de las personas necesarias. https://pmi-

mad.org/socios/articulos-direccion-proyectos/691-la-medicion-total-del-tamano-de-los-proyectos-software-y-sus-beneficios-para-la-gestion-del-proyecto

Weltanschauung: Una cosmovisión, calco del alemán Weltanschauung, es una imagen o figura general de la existencia, realidad o mundo que una persona, sociedad o cultura se forman en una época determinada; y suele estar compuesta por determinadas percepciones, conceptuaciones y valoraciones sobre dicho entorno. Dilthey, W. (1914). «Einleitung in die Geisteswissenschaften. Versuch einer Grundlegung für das Studium der Gesellschaft und der Geschichte». Leinen: Vandenhoeck & Ruprecht. ISBN 3-525-30301-7.

Referencias

[1] Lomprey, Gérald. Hernandez, Saulo. La importancia de la calidad en el desarrollo de productos de software. Technical Report COMP-018-2008. Universidad de Montemorelos, México. Recuperado de: http://fit.um.edu.mx/CI3/publicaciones/Technical%20Report%20COMP-018-2008.pdf

[2] Marc Hamilton in conjunction with Harris Kern's Enterprise Computing Institute. Organizing for Successful Software Development

[3] Boehm, Barry. The future of Software and Systems Engineering Processes. University of Southern California, Los Angeles, CA. 2005

[4] Checklannd, Peter. Scholes, Jim. La metodología de los sistemas suaves de acción. WILEY. México, DF. 1994.

[5] Nirenberg, Olga. Brawerman, Josette. Ruiz, Violeta. Evaluar para la Transformación. Innovaciones en la evaluación de programas y proyectos sociales. PAIDÓS TRAMAS SOCIALES. Argentina, 2000.

[6] Capability Maturity Model. Recuperado de: https://es.wikipedia.org/wiki/Capability_Maturity_Model

[7] Capability Maturity Model Integration. Recuperado de: https://es.wikipedia.org/wiki/Capability_Maturity_Model_Integration

[8] Qué es una fábrica de software, Jumay. Ago 2017. Recuperado de: http://jumay.biz/articulos/desarrollo-de-software/quees-fabrica-software/

[9] El desarrollo de software. Complejidad y Tecnologías de la Información (Tecnologías de la información). Recuperado de: http://dit.upm.es/~fsaez/intl/libro_complejidad/15-el-desarrollo-del-software.pdf

[10] Servicios integrales en computación. Recuperado de: http://www.sincows.com/sincows/index.php?option=com_content&view=article&id=70&Itemid=68

[11] Diagnóstico del sistema de administración tecnológica. Comisión Federal para la Protección contra riesgos Sanitarios. Informe Final. Estado de México, 2011.

[12] Díaz Fidencio, Rodríguez Gregorio, Valencia Alejandro, Saldaña Héctor, González Miguel. Ciencias Administrativas y Sociales Handbook T-II: Congreso Interdisciplinario de Cuerpos Académicos / coord. por Teresa Ramírez Cano,

Patricia del Carmen Mendoza García, 2013, ISBN 978-607-8324-08-8, págs. 140-148

[13] AYESA. Recuperado de: https://www.ayesa.com/es/prensa/noticias/599-ayesa-alcanza-el-maximo-nivel-de-cmmi-para-desarrollo-de-software-internacional

[14] Stacks alcanza el nivel 2 de CMMI en su Barcelona Health Factory Software. Recuperado de: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=17&cad=rja&uact=8&ved=2ahUKEwj6urPalv3oAhUCTKwKHaT0D3E4ChAWMAZ6BAgHEAE&url=http%3A%2F%2Fwww.stacks.es%2Fdocuments%2F14771%2F9a78915f-0042-4a9a-a41e-26c4fe74ad96&usg=AOvVaw14HEZsMVG2atnL92AbmuOf

[15] ¿Cómo determinar el tamaño de una muestra? Recuperado de: https://www.psyma.com/company/news/message/como-determinar-el-tamano-de-una-muestra

[16] Población y muestra. Revisión científica por Ana Zita. Doctora en Bioquímica. Escrito por Zara Lugo. Recuperado de: https://www.diferenciador.com/poblacion-y-muestra/#:~:text=La%20muestra%20es%20una%20parte,que%20se%20puede%20contabilizar%20f%C3%A1cilmente.