Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223893475
RFC: ATI120618V12

Revista Dilemas Contemporáneos: Educación, Política y Valores.
http://www.dilemascontemporaneoseducacionpoliticayvalores.com/
Año: VII Número: Edición Especial Artículo no.:60 Período: Noviembre, 2019.

TÍTULO: La gestión de la calidad en las factorías de software: una alternativa eficaz.

AUTORES:

1. Máster. Segobia Ocaña Manuel Alberto.

2. Máster. Torres Vargas Richard Javier.

3. Máster. Sobenis Cortez Juan Alipio.

RESUMEN: Existen cuantiosas empresas que operan en la industria del software, pero no todas brindan a los clientes un alto grado de satisfacción; la factoría de software es una alternativa para enfrentar las insuficiencias, al lograr una colocación estructurada de elementos que componen el proceso de desarrollo de sistemas informáticos. Un componente que contribuye a elevar la calidad de los procesos y productos informáticos, enfoque de gestión de la calidad durante el proceso de desarrollo. El propósito de este trabajo es presentar un procedimiento para la gestión de la calidad en las factorías de software que contribuya a la mejora de la satisfacción de los requerimientos de los clientes e incluye un tratamiento a los riesgos para minimizar su ocurrencia.

PALABRAS CLAVES: factoría de software, gestión de la calidad, gestión de riesgos.

TITLE: Quality management in software factories: an effective alternative.

AUTHORS:

1. Máster. Segobia Ocaña Manuel Alberto.

2. Máster. Torres Vargas Richard Javier.

3. Máster. Sobenis Cortez Juan Alipio.

ABSTRACT: There are many companies that operate in the software industry, but not all offer customers a high degree of satisfaction, the software factory is an alternative to address the shortcomings, achieving a structured placement of elements that make up the process of developing computer systems; a component that contributes to raise the quality of computer processes and products, quality management approach during the development process. The purpose of this work is to present a procedure for quality management in software factories that contributes to improving the satisfaction of customer requirements and includes a treatment of risks to minimize their occurrence.

KEY WORDS: software factory, quality management, risk management.

INTRODUCCIÓN.

La sociedad actual ha sufrido cambios que se han visto reflejados en la rápida evolución del hardware de computadoras, lo cual ha provocado un impacto en el hombre, que experimenta percepciones diferentes según los tipos de plataformas y software de aplicaciones de las que disponen las organizaciones a las que se vincula. Para el grupo de personas que se encarga de desarrollar los softwares, también constituye un reto satisfacer a los clientes en materia de productos informáticos. Se requiere personal capacitado y grandes esfuerzos para ser administrado y desarrollado; de organización y disciplina para obtener lo que el mercado demanda, que se traduce en un producto con calidad; que esté disponible en tiempo y presupuesto y pueda absorber los cambios que la creciente evolución tecnológica demanda.

Existen numerosas empresas que han incursionado en la industria del software, pero lamentablemente no todas brindan a los clientes un alto grado de satisfacción. En muchos casos, esta situación depende en gran medida de la fragilidad que existe entre la relación que tiene la planificación inicial y el tiempo real que se emplea en el desarrollo de los proyectos, la desorganización del trabajo, la no utilización de estándares de calidad, la elección incorrecta de la metodología de desarrollo de software a utilizar, falta de capacitación del personal, y como resultado, la mala calidad de los productos. Uno de los elementos que puede contribuir considerablemente a elevar la calidad de los procesos y productos es el empleo del enfoque de gestión de la calidad durante el proceso de desarrollo de software.

Cusumano (1992) señala, que "El desarrollo de una factoría implica que las buenas prácticas de Ingeniería de Software sean aplicadas sistemáticamente". Otros criterios similares (Gamboa, 2014; Rozo, 2014; Martínez, Arango y Robledo, 2015) consideran que la factoría de software debe tener bien definidos los roles del equipo de desarrollo y personal especializado, lo que promueve un proceso normalizado, repetible y mejorable continuamente, mediante actividades de desarrollo predecibles y de reducción de los riesgos; para que queden definidas las responsabilidades, y concentrar sus esfuerzos en un área determinada de la producción.

Según lo expuesto anteriormente y con la intención de aportar nuevos resultados científicos acerca del tema en cuestión; el propósito de este trabajo consiste en desarrollar un procedimiento para la gestión de la calidad en las factorías de software que contribuya a la reducción de los riesgos durante el proceso de desarrollo y a la satisfacción de los requerimientos de los clientes. Se encuentra distribuido en tres secciones; la primera, "las factorías de software, una solución inteligente", destaca la importancia de estas para la mejora en el desarrollo de productos informáticos; la segunda, "la gestión de la calidad en las factorías de software", enfatiza la importancia de gestionar la calidad y subraya la gestión de riesgo como elemento esencial en el proceso de desarrollo; y la tercera,

"procedimiento para la gestión de la calidad en la factoría de software", presenta una propuesta a tener en cuenta para gestionar la calidad en cada una de las etapas del ciclo de vida de un producto informático.

DESARROLLO.

Las factorías de software, una solución inteligente.

Se le denomina factoría a cualquier fábrica o industria, en la cual se lleve a cabo la transformación de materias primas en otros productos. Por extensión se aplica esta palabra para designar determinadas actividades en las que no se produce uso y transformación de elementos y que tienen como objeto final la obtención de productos intangibles: factoría de comunicación, de cine y de software (Bagarotti y Ávila, 2012).

Otro criterio de interés es el de López, André e Infante (2011), al considerar que una factoría de software es una organización que posee procesos estructurados y mejorados de forma continua, dirigida a la creación de productos de software, acorde con los requerimientos documentados de usuarios y clientes, de la forma más productiva y económica posible.

Las factorías de software se perfeccionan en la unión del conocimiento y la metodología, en la que se acumule todo lo desarrollado, lo que permite conseguir altos porcentajes de reutilización. La industrialización del proceso de software facilita la evaluación, medición y control, y con ello, su mejora y adaptación al cambio, no solo en el análisis de los asuntos internos, sino en la investigación de nuevas tecnologías, herramientas y métodos (Pons, 2012 y López, 2007).

Otro elemento imprescindible que se debe conocer es cómo llevar a cabo la implantación de una factoría de software. En relación con el tema, Garzás y Piattini (2007) proponen el uso de la Metodología Unificada de Factoría de Software (MUFS), quienes señalan que es un camino ordenado, conformado por un conjunto de pasos conducentes hacia el desarrollo de software en el contexto de

factoría. Consideran las interrelaciones existentes entre marco teórico y métodos, conocimiento del objeto, y finalmente, la relación entre métodos y objeto.

Después de haber analizado las definiciones y anotaciones ofrecidas por varios autores, se asume por los articulistas que una factoría de software es la colocación estructurada de todos los elementos que componen el proceso de desarrollo de software (figura 1), promueve el uso de procesos estandarizados y repetibles, fuerte comunicación con el cliente, alta calidad de los procesos y productos, estricta y continua capacitación de sus recursos humanos con roles y responsabilidades especialmente definidos y enfocados en la concentración de los esfuerzos en un área determinada de la producción.

Figura 1. Representación simple de la factoría de software. **Fuente:** elaboración propia.

Una de las características fundamentales de los proyectos a desarrollar por los miembros de la factoría de software es su dimensión temporal, diferente a los fenómenos de producción inmediata, lo que demanda de un tiempo importante para su realización. Esta característica temporal hace que un

proyecto debe concebirse como un proceso, es decir, como una sucesión de actividades, que en muchos casos pueden superponerse durante el tiempo de ejecución.

Los proyectos se caracterizan por tener un principio y un fin (Flores, Paiva y Letra, 2016). Hacia su interior se encuentran partes o componentes tradicionales de la administración, como planificación, ejecución y control; de este modo, se puede comprender por procesos tradicionales de la administración, la secuencia de actividades que ocurren desde el principio, pasa por la planificación, la ejecución y el control, y finalizan con el cierre del proyecto.

Uno de los errores más comunes que se comete a la hora de planificar y ejecutar un proyecto de software es ordenarlos en forma de 'vagones de tren' de manera secuencial, puesto que la calidad solo puede ser controlada al final del proceso. Por este motivo se propone un aprovechamiento mejorado en el uso del tiempo a través de la superposición de las actividades, como se propone en la figura 2, en la que se incluyen elementos de control de la calidad en todo momento del desarrollo del proceso.

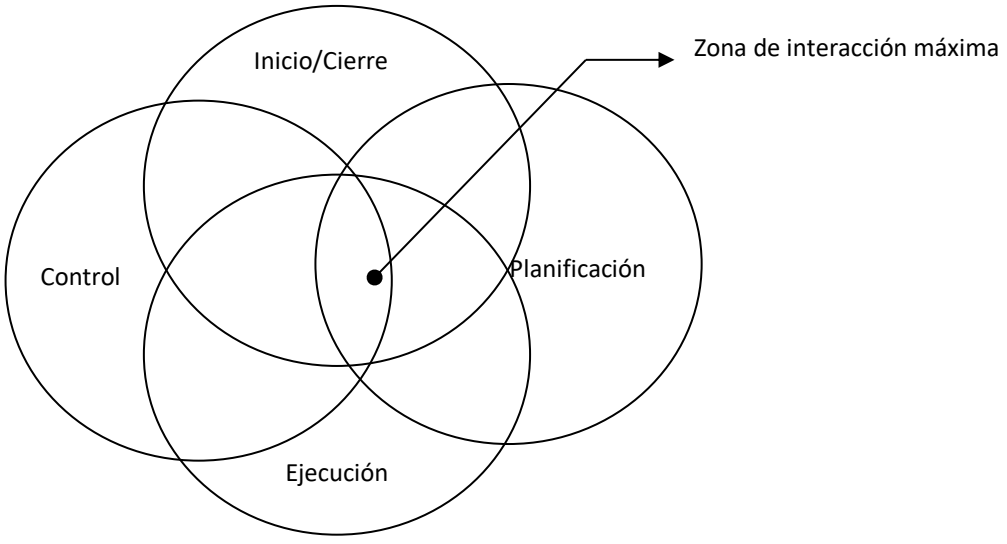

Figura 2. Interacción mejorada de los procesos en la factoría de software. **Fuente:** elaboración propia.

La gestión de la calidad en las factorías de software.

Las organizaciones a nivel internacional reconocen que la calidad del producto se traduce en ahorro de costos y en una mejora general. La industria de software se suma a ello, por lo que en los últimos años se han realizado intensos trabajos para aplicar los conceptos de calidad en el ámbito del software. El concepto de calidad de software, según Pressman (2010), se asocia con la relación de los requisitos funcionales y de rendimiento explícitamente establecido mediante los estándares de desarrollo plenamente documentados y las características implícitas del producto final. De este planteamiento se definen los elementos esenciales en el desarrollo del software; de no cumplirse, se consideran no conformidades que atentan contra la calidad.

- Los requisitos del software, los cuales son definidos por el cliente y constituyen los requerimientos básicos que se traducen en medidas para la evaluación de la conformidad final.
- Los estándares de desarrollo que acotan los criterios que guían la forma en que se aplica la ingeniería del software o del conocimiento.

La calidad del software va a depender en gran medida de requerimientos o requisitos. Estos deben ser claramente entendidos por todos y acotados a la hora de solicitar un servicio de desarrollo de software, considerados la máxima expresión de las necesidades de los usuarios. La calidad, como objeto de gestión de una organización, necesita definir estos procesos y medirlos para poder gestionarlos; es decir, para tener la capacidad de proponer mejoras y reconocerlas.

Existe un vínculo estrecho entre la calidad de proceso de desarrollo y la calidad de los productos desarrollados que utilicen dicho proceso (Cruz, López y Ruiz, 2017). En consecuencia, muchas organizaciones de ingeniería de software han tomado el camino de la mejora de procesos del software para optimizar sus resultados (Sommerville, 2005). Resulta necesario identificar productos de calidad, examinar el proceso utilizado para desarrollarlos y generalizar esos procesos para aplicarlos a otros proyectos.

La gestión de riesgos es parte necesaria en la gestión de la calidad de un producto informático. Uno o varios riesgos pueden poner en peligro la viabilidad de un proyecto cualquiera debido al impacto que puede causar en estos (Mera, 2016). Por esta razón, es importante estar preparados y trabajar de forma proactiva mediante la identificación y mitigación de los riesgos que se puedan presentar.

El análisis y la gestión del riesgo son una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre. Un proyecto de software puede estar lleno de problemas; un riesgo es un problema potencial −puede ocurrir o no−; pero sin tener en cuenta el resultado, realmente es una buena idea identificarlo, evaluar su probabilidad de aparición, estimar su impacto, y establecer un plan de contingencia por si ocurre el problema (Pressman, 2010).

Según la *Guía avanzada de gestión de riesgos*, emitida por el Instituto Nacional de Tecnologías de la Comunicación (INTECO, 2008), el riesgo del proyecto es un evento o condición incierta que, si se produce, tendrá un efecto negativo sobre al menos un objetivo del proyecto, como tiempo, costo, alcance o calidad.

Aunque hay varias definiciones para riesgo de software se concuerda con lo plateado por Pressman (2010), quien considera que el mismo siempre implica dos características:

- Incertidumbre (probabilidad): el acontecimiento que caracteriza al riesgo. Puede o no puede ocurrir; por ejemplo, no hay riesgos de un 100 % de probabilidad.
- Pérdida (impacto): si el riesgo se convierte en una realidad ocurrirán consecuencias no deseadas o pérdidas.

La gestión de riesgos se lleva a cabo:

- En la elaboración de una propuesta, cuando se planifica el proyecto.
- A intervalos regulares durante la vida del proyecto: por ejemplo, como parte de los informes de su cumplimiento.
- Cuando hay un cambio de alcance.

Por tanto, es un proceso iterativo y recurrente a lo largo de toda la vida del proyecto. El propósito de la gestión de riesgos es minimizar la probabilidad y consecuencia de amenazas identificadas, de tal forma que los objetivos del proyecto se cumplan. Esto se logra a través del seguimiento de una serie de pautas:

- Identificar todos los riesgos conocidos del proyecto.
- Realizar una evaluación de la probabilidad de ocurrencia y del impacto potencial.
- Cuantificar cuál sería el costo de los riesgos en caso de que ocurrieran.
- Crear planes de acción para gestionar los riesgos de alta prioridad.
- Reconocer y gestionar los riesgos lo antes posible.

Al gestionar los riesgos de forma efectiva, se conseguirá mejorar el beneficio de la factoría de software, lo que favorecerá el cumplimiento de los objetivos generales de la organización y los propios del proyecto; de esta manera se evitarán problemas que pudieran causar pérdidas inesperadas y no planificadas. Otros beneficios que se obtendrán al llevar a cabo una buena gestión de los riesgos será la reducción de los costos del proyecto y la mejora de la satisfacción del cliente, lo que incrementará la capacidad y probabilidades de éxito, la facilidad del desarrollo del proyecto y la disminución drástica de las sorpresas en los proyectos (INTECO, 2008).

Si un equipo de software adopta un enfoque proactivo frente al riesgo, evitarlo es siempre la mejor estrategia. Esto se consigue mediante el desarrollo de un plan de reducción del riesgo. A medida que progresa el proyecto comienzan las actividades de supervisión del riesgo, según factores que pueden proporcionar una indicación sobre si el riesgo se hace más o menos probable.

Procedimiento para la gestión de la calidad en la factoría de software.

Hasta este momento es evidente que la gestión de la calidad no es un hecho fortuito sino debido a un proceso continuo, compuesto por epatas y tareas que aseguren en cada momento la calidad; en

consecuencia, se plantea por los autores un procedimiento que incluye elementos relacionados con la gestión de los riesgos dentro del desarrollo del software y que asegure cada requerimiento en las etapas del proceso. En la figura 3, se describen tres epatas fundamentales, asociadas a los tres momentos más importantes en la vida del software: al principio del proyecto, durante el desarrollo y al finalizar; así como una serie de acciones de relevancia a realizar en cada una de ellas.

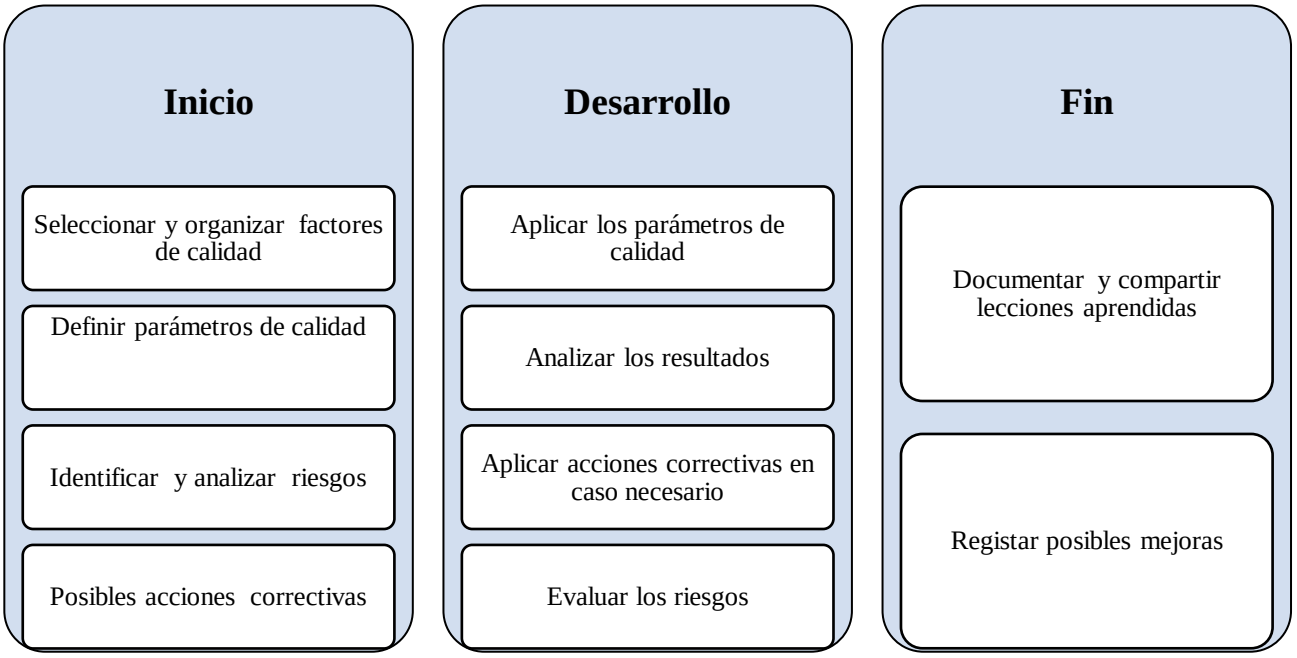

Figura 3. La gestión de la calidad de software en tres etapas. **Fuente:** elaboración propia.

El propósito de la primera etapa “Inicio” es definir los factores de calidad del proyecto, que tienen en cuenta los requisitos del cliente y los estándares de desarrollo. Para ello es necesario tener presente la relación que poseen los factores con las características peculiares del producto o proyecto, los que dependen en gran medida de los requerimientos no funcionales y de las restricciones que puede tener el proyecto. Así, por ejemplo, si se espera que el ciclo de vida del sistema sea largo, la ‘facilidad de mantenimiento’ y la ‘flexibilidad’ se convierten en un requisito; si el sistema es experimental y se espera que las especificaciones del sistema cambien frecuentemente, la ‘flexibilidad’ será importante y sin embargo la ‘eficiencia’ apenas tendrá importancia; si el sistema se desarrolla para un entorno

en el que el hardware evoluciona rápidamente, la ‘portabilidad’ es esencial; si se espera que ciertas funciones del sistema, se utilicen por un largo periodo de tiempo, aunque el resto del sistema cambie, la ‘facilidad de reutilización’ será fundamental, etc. Estos factores de calidad se convierten en los parámetros de calidad a tener en cuenta durante todo el ciclo de desarrollo del software. Posteriormente, se organizan en orden de importancia por parte del equipo de desarrollo.

De acuerdo con las características del sistema informático, se conforma una lista de chequeo que proporciona el seguimiento en cada momento, según los valores deseables establecidos en función de datos históricos o referencias externas. A continuación, se definen las diferentes funciones de cada uno de los miembros del equipo, las que incluye quién va a medir los parámetros de calidad y en qué momento.

La identificación de los riesgos es un elemento a tener en cuenta en esta etapa. De manera sistemática e iterativa, a lo largo del proyecto se determinan cuáles son los riesgos que puedan afectar y se documentan sus características. Lo primero a tener en cuenta es identificar las categorías de riesgos: técnicos, del negocio, conocidos, predecibles, impredecibles, entre otros. Para cada una de estas categorías existen los riesgos genéricos (son una amenaza potencial para todos los proyectos de software) y los específicos (solo lo pueden identificar las personas que tienen un amplio conocimiento dentro del equipo de trabajo).

Los síntomas de riesgo o señales de aviso son indicadores de que un riesgo ha ocurrido o está a punto de ocurrir, y por lo tanto es necesario buscar una solución. Los disparadores pueden ser eventos específicos, un ejemplo puede ser no cumplir ciertos hitos relevantes, lo que implica un inminente retraso en la agenda programada. Estos disparadores también pueden ser umbrales predefinidos como, por ejemplo, que una estimación en el desarrollo de un producto exceda el 10 % de lo planificado en las primeras cuatro semanas, lo que indica que se ha detectado un riesgo. En ocasiones puede ser útil establecer disparadores tempranos que proporcionen tiempo suficiente para tratar los riesgos

inminentes. Estos disparadores, al igual que los riesgos, serán controlados y revisados como parte del proceso de control y monitoreo.

Los riesgos deben ser registrados y analizados como parte de su proceso de gestión y las respuestas correctivas constituyen una forma de mantener el control sobre el desarrollo del software. Se propone la ficha que se muestra en la tabla 1, la que debe ser rellenada para cada riesgo.

En el análisis de riesgos futuros (cualitativo o cuantitativo) se evalúa la probabilidad de que ocurra el impacto y la prioridad de cada uno. La mejor práctica para llevar a cabo el análisis cualitativo es la de utilizar un conjunto de valores fijos en todos los proyectos que representen la probabilidad y el impacto de cada uno desde un punto de vista cualitativo.

Tabla 1. Ficha de riesgo.

Riesgo	**Nominativo**
Estado actual y descripción	(Debe incluir el evento, el momento en que ocurrió y el impacto)
Identificado	(Ha sido identificado, pero no ha sido analizado ni evaluado)
Evaluado	(identificado que actualmente no tiene un plan de respuesta)
Planificado	(identificado y cuenta con un plan de respuesta)
En proceso	(La respuesta al riesgo está en ejecución)
Cerrado	(Ocurrió y que ha sido cerrado
No ocurrido	(Fue identificado, pero que no ocurrió porque fue gestionado antes de que sucediera)

Fuente: elaboración propia.

Estos valores servirán para categorizar, agruparlos y proporcionar una guía sobre dónde invertir el mayor esfuerzo. Si por el contrario cada equipo eligiese sus propios valores, no existiría una base común y no se podría, por ejemplo, comparar riesgos de forma efectiva, ni determinar cómo mejora la factoría de software en su gestión. Para cada riesgo identificado se evalúan la probabilidad y el impacto, es decir, se asocia al riesgo un valor cualitativo de probabilidad e impacto. El impacto podría afectar al costo, la planificación, la calidad o una combinación de los anteriores. El equipo debería definir no solo la magnitud del impacto, sino también cuál es el elemento del proyecto más afectado.

Al estimar la probabilidad de ocurrencia del riesgo, el jefe de proyecto y el equipo los analizan para determinar la probabilidad de que ocurrencia.

Aunque es importante identificar el mayor número posible de riesgos del proyecto, en muchos casos el número identificado puede ser abrumador, y lógicamente el equipo de trabajo no podrá realizar un seguimiento ni una gestión efectiva de todos ellos. Una solución sería agruparlos en función de sus prioridades, de tal forma que el equipo pueda centrarse en los más críticos.

La evaluación de la importancia de cada riesgo, y por consiguiente, de su prioridad, generalmente se realiza mediante el uso de una matriz de probabilidad e impacto. Esta matriz asignará categorías a cada uno, sobre la base de la combinación de dichos factores (probabilidad e impacto) que llevan a la calificación de los riesgos como de prioridad baja, moderada o alta. Pueden usarse términos descriptivos o valores numéricos, lo que depende de la preferencia de la organización.

En el análisis cuantitativo de los riesgos es de gran ayuda la evaluación matemática de la probabilidad de ocurrencia de cada uno y sus consecuencias en las salidas del proyecto. Para determinar el impacto es necesario analizar los efectos del riesgo en los componentes del proyecto: alcance, calidad, planificación y costo del proyecto; y evaluarse mediante la utilización de los datos numéricos. Igualmente, la determinación de la probabilidad de ocurrencia del riesgo analizado se ubica en el cuadrante correspondiente. Una vez que se define la probabilidad e impacto de cada uno, es útil hacer una tabla para registrarlos.

En la determinación de las posibles acciones correctivas es necesario planificar la respuesta a los riesgos identificados con la intención de logar evitarlos, supervisarlo y gestionarlo. El equipo de software adopta un enfoque proactivo frente al riesgo; evitarlo es siempre la mejor estrategia. Esto se consigue con el desarrollo de un plan de reducción del riesgo. A medida que progresa el proyecto, comienzan las actividades de supervisión. El jefe del proyecto supervisa factores que pueden proporcionar una indicación sobre si el riesgo se hace más o menos probable. Además de supervisar

los factores apuntados anteriormente, el jefe del proyecto debería supervisar también la efectividad de los pasos de reducción del riesgo. La gestión del riesgo y los planes de contingencia asumen que los esfuerzos de reducción han fracasado y que el riesgo se ha convertido en una realidad.

La segunda etapa del procedimiento, "desarrollo", tiene como objetivo evaluar la calidad del software de acuerdo con los parámetros definidos, monitorizar los riesgos identificados y proporcionarles evidencias a los clientes del desarrollo del producto. Tiene lugar la toma de medidas para corregir las desviaciones de los parámetros evaluados. Se incluye también la ejecución de los planes de reducción de riesgos a lo que se asocia monitorizar, revisar y actualizar el estado de estos y los planes de respuesta; además de identificar nuevos riesgos.

Por su parte, la etapa "fin" tiene como objetivo documentar las lecciones y registrar posibles mejoras. En ella ocurre la generalización de aspectos positivos de las listas de comprobación y las medidas de la gestión de riesgos a todo el proyecto. Compartir estas lecciones es un recurso muy valioso en el ámbito de la gestión de riesgos; se organiza como parte del proceso de informes del cierre del proyecto y se comparte con el resto de profesionales de la factoría de software. Propicia experiencia general sobre el proceso de gestión de la calidad de software y de riesgos, y su relación con la satisfacción del cliente.

CONCLUSIONES.

El modelo de factoría de software enfatiza las ventajas de un nuevo enfoque para desarrollar software, al lograr que se promueva el uso de procesos estandarizados y repetibles; implica mejorar la organización de los miembros del equipo y la asignación de roles de la manera más eficiente posible. La gestión de la calidad en el proceso de desarrollo de productos informáticos contribuye al cumplimiento de los requisitos de los clientes y los estándares de desarrollo. El control de los riesgos

conseguirá mejorar el beneficio de la factoría de software y favorecerá el cumplimiento de los objetivos generales de la organización y los propios de los proyectos.

Con la propuesta metodológica expuesta se gestiona la calidad del sistema informático en la factoría de software, y se definen los elementos esenciales para medir su calidad y realizar la completa gestión de los riesgos informáticos asociados.

REFERENCIAS BIBLIOGRÁFICAS.

1. Bagarotti, Y. y Ávila, A. (2012). Las factorías de software, una alternativa para garantizar la calidad de los productos. Ingeniería UC, 19(1), pp.1–13.

2. Cruz, F. L., López, A. y Ruiz, C. (2017). Sistema de Gestión ISO 9001-2015: técnicas y herramientas de ingeniería de calidad para su implementación. Ingeniería Investigación y Desarrollo, 17(1), pp.59 – 69.

3. Cusumano, M. A. (1992). Shifting economies: From craft production to flexible systems and software factories. Research Policy, 21(5), pp.453 – 480.

4. Flores, N. H., Paiva, A. C. R. y Letra, P. (2016). Software Engineering Management Education through Game Design Patterns. Social and Behavioral Sciences, 228(2), pp.436 – 442.

5. Gamboa, J. (2014). Aumento de la productividad en la gestión de proyectos, utilizando una metodología ágil aplicada en una fábrica de software en la ciudad de Guayaquil. Revista Tecnológica ESPOL–RTE, 27(2), pp.1–36. Recuperado de: http://www.rte.espol.edu.ec/index.php/tecnologica/article/view/312

6. Garzás, J. y Piattini, M. (2007). Concepto y evolución de las fábricas de software. Kybele Consulting. Recuperado de: https://studylib.es/doc/7145744/concepto-y-evoluci%C3%B3n-de-las-fabricas-software

7. INTECO. (2008). Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España. Laboratorio Nacional de Calidad del Software. España.

8. López, Y. (2007). Modelo de factoría de software aplicando inteligencia. Revista Cubana de Ciencias Informáticas, 1(1), pp.139–168.

9. López, Y., André, M. e Infante, A. L. (2011). Formación de roles y buenas prácticas en el trabajo por la calidad de un ingeniero informático. Ingeniare, Revista Chilena de ingeniería, 19(3), pp.382–395. Recuperado de: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-33052011000300008

10. Martínez, S. J., Arango, S. y Robledo, J. (2015). El crecimiento de la industria del software en Colombia: un análisis sistémico. Revista IEA, 12(23), pp.95 –106.

11. Mera, J. A. (2016). Análisis del proceso de pruebas de calidad de software. Ingeniería Solidaria, 12(20), pp.163–176. Recuperado de: https://revistas.ucc.edu.co/index.php/in/article/view/1482

12. Pons, N. L. (2012). Definición de la Entidad Inteligencia Empresarial para un Modelo de Factoría Aplicando Inteligencia. Cuba. RECAI, 1(2), pp.41–57.

13. Pressman. R. (2010). Ingeniería del Software. Un enfoque práctico. España: Ed: McGraw-Hill Interamericana.

14. Rozo, J. (2014). Metodología de Desarrollo de Software: MBM. Ingeniare, Universidad Libre-Barranquilla, 9(16), pp.111-125.

15. Sommerville, I. (2005). Ingeniería de Software. (7.ª ed.). Pearson educación, S.A.

DATOS DE LOS AUTORES.

1. **Manuel Alberto Segobia Ocaña.** Magister en Conectividad y Redes de Ordenadores. Docente de la Universidad Técnica de Babahoyo – Ecuador. E-mail: msegobia@utb.edu.ec

2. **Torres Vargas Richard Javier.** Magister en Ingeniería y Sistemas de Computación. Docente agregado de la Universidad Técnica de Babahoyo – Ecuador. E-mail: rtorres@utb.edu.ec
3. **Sobenis Cortez Juan Alipio.** Magister en Gerencia De Proyectos Educativos y Sociales, Docente de la Universidad Técnica de Babahoyo – Ecuador. E-mail: jsobenis@utb.edu.ec

RECIBIDO: 9 de octubre del 2019. **APROBADO:** 18 de octubre del 2019.