

Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico

Quality assurance method in a software development methodology: a practice approach

Dante Carrizo^{1*} Andres Alfaro²

Recibido 24 de octubre de 2016, aceptado 25 de septiembre de 2017

Received: October 24, 2016 Accepted: September 25, 2017

RESUMEN

El Aseguramiento de la Calidad de Software (ACS), es un conjunto de métodos, herramientas y técnicas que permiten gestionar la calidad en el desarrollo de un producto de software. Pese a ser un elemento fundamental a la hora desarrollar un proyecto, no todas las empresas lo aplican debido a presupuesto, falta de personal o adaptaciones de estándares más complejos. Este artículo, presenta un enfoque práctico como guía estratégica, para administrar la calidad en el desarrollo de un proyecto de software. Para esto, se presenta un método de ACS que consta de tres componentes. La Esencia, que busca que todo el equipo de trabajo entienda el concepto de calidad; que no solo se ve reflejado en actividades o tareas, sino también en la forma cómo trabaja el equipo. Herramientas, que tienen como finalidad controlar la calidad en el proyecto de software. Y por último, las Métricas utilizadas no solo para medir los resultados obtenidos, sino también para poder mejorar los procesos internos. El Método fue aplicado en un caso de estudio, para el desarrollo de una aplicación web, ayudando a controlar de mejor manera los cambios y futuros riesgos que podrían ocurrir en el desarrollo del proyecto, proporcionando una forma de trabajo más segura. Aunque el método facilitó satisfacer las necesidades del cliente, no se ha probado en equipos de trabajo de mayor tamaño. Además, la recolección de métricas es a largo plazo, lo que para este estudio de caso, no pudo ser probada.

Palabras clave: Calidad del software, aseguramiento de calidad, enfoque práctico, caso de estudio, emparrillado.

ABSTRACT

Software Quality Assurance (SQA) is a set of methods, tools and techniques that allows managing the quality in the development of a software product. In spite of a fundamental element in developing a project, not all companies apply due to budget matter, lack of staff or more complex standards adaptations. This article, presents a practical approach as a strategic guide, to manage the quality in the development of a software project. For this, a method of SQA is presented that consists of three components: The Essence, which seeks that work team understands the concept of quality; which is not only reflected in activities or tasks, but also in the way the team works. Tools, whose purpose is to control the quality in the software project. Finally, the Metrics used not only to measure the results obtained, but also to improve internal processes. The Method was applied in a case study, for the development of a web application, helping to

¹ Departamento de Ingeniería Informática y Ciencias de la computación. Universidad de Atacama. Casilla 240. Copiapó, Chile. dante.carrizo@uda.cl

² Departamento de Ingeniería Informática y Ciencias de la computación. Universidad de Atacama. Casilla 240. Copiapó, Chile. andres.alfaro@uda.cl

* Autor de correspondencia

better control the changes and future risks that could occur in the development of the project, providing a more secure way of working. Although the method facilitated meeting customer needs, it has not been tested in larger workpieces. In addition, the collection of metrics is long-term, which for this case study, could not be proven.

Keywords: Software quality, quality assurance, practical approach, case study, repertory grid.

INTRODUCCIÓN

La búsqueda por entregar productos y servicios que satisfagan las necesidades de los clientes, es el objetivo de todas las empresas que quieren posicionarse en los diferentes mercados. Las empresas desarrolladoras de software no escapan a esta premisa. Pero, ¿Cómo se puede satisfacer las necesidades del cliente, en un desarrollo de un producto de software? Esto se puede lograr a través de la calidad. En términos generales, la calidad busca que el cliente quede satisfecho y conforme con su producto.

La calidad en ingeniería del software es el cumplimiento de los requerimientos contractuales por parte del producto software desarrollado, así como durante el proceso de desarrollo. La calidad se obtiene mejorando día a día el proceso de producción, mantenimiento y gestión del software [1]. Para optimizar la calidad de los productos y/o servicios es preciso conocer al cliente y sus necesidades, conocer la competencia y poseer un modelo de calidad. Esto último permitirá incrementar la fiabilidad, reducir el mantenimiento, aumentar la satisfacción del cliente, mejorar la dirección del proyecto, detectar errores lo más temprano posible e incrementar el beneficio para el desarrollador. La función de aseguramiento de la calidad del software (ACS) se basa en un planificado y sistemático diseño de acciones y métodos requeridos para garantizar la calidad del mismo [2]. El alcance de la responsabilidad del aseguramiento de la calidad, en el desarrollo de software, abarca muchos constituyentes de una organización, tales como: ingenieros de software, desarrolladores, líderes de proyecto, clientes y personas encargados se ACS.

La calidad es importante en el desarrollo de un producto o servicio y, más aún, en la creación de un producto de software, no solo porque busca cumplir con las expectativas del cliente, sino también por mejorar los procesos internos en la elaboración de

un producto, tarea fundamental en el crecimiento y posicionamiento de una empresa.

En la actualidad se ha desarrollado un conjunto de modelos, estándares y metodologías como CMM, ISO/IEC 15504 [3] que ayudan a las organizaciones a mejorar la calidad de sus procesos y productos, así como sus proyectos. No obstante, debido a las características particulares de la pequeñas empresas desarrolladoras, tales como: presupuestos limitados, carencia de personal especializado, infraestructura inadecuada así como reservas financieras limitadas, complejidad en los proyectos y competidores, en su afán de sobrevivir toman el camino más corto disponible, el cual no necesariamente está en armonía con las mejores prácticas de aseguramiento de la calidad [4-5].

El establecimiento de estos modelos y metodologías, dado el contexto de las pequeñas empresas, debiera realizarse de un modo particular y visiblemente diferente a como se hace en organizaciones de mayor tamaño. Además esto no es tan sencillo como el hecho de considerar los programas de mejora como versiones a escala de las grandes compañías [6], ya que las pequeñas empresas desarrolladoras de software requieren prácticas eficientes de Ingeniería de Software adaptadas a su tamaño y tipo de negocio [7].

Es por ello, que en este trabajo se propone un método de Aseguramiento de la Calidad del Software, que se puede unir o acoplar a cualquier metodología de desarrollo. Además, de no ser necesario, un equipo de ACS, por lo tanto, no aumentan los costos y a la vez no interfiriendo o modificando mucho la forma de trabajo del equipo.

Este trabajo está compuesto por una recolección de antecedentes acerca de la calidad, revisión de trabajos relacionados, para luego, mostrar la propuesta metodológica de aseguramiento de la calidad, y finalmente, aplicarla al caso de estudio.

ANTECEDENTES

En [8], se define a la ingeniería de software como una tecnología con varias capas, como se observa en la Figura 1, donde las herramientas son la ayuda automatizada para el proceso y los métodos. A su vez los métodos son actividades técnicas requeridas para creación de productos de software, abarcando una gran gama de tareas que incluyen análisis de requisitos, diseño, programación, test y mantenimiento. Como marco de trabajo, que ayuda a controlar la gestión del proyecto y las actividades de ingeniería e identifica las tareas que se deben realizar para construir un software de alta calidad esta la capa de proceso. Y por último, la capa que da soporte a la ingeniería de software es el compromiso con la calidad.



Figura 1. Capas de la Ingeniería de Software en [8].

Desde la capa de Compromiso con la Calidad, se entiende que la calidad es fundamental para la ingeniería de software, y, por lo tanto, para la creación de un producto de software. Es decir, la calidad es una actividad sombrilla que se aplica a todo el proceso de software.

Para definir el concepto de aseguramiento de la calidad es necesario conocer la evolución y el desarrollo de la calidad, y más específicamente calidad del software.

En Ishikawa [9], se define calidad como “Desarrollar, diseñar, manufacturar y mantener un producto de calidad que sea el más económico, el útil y siempre satisfactorio para el consumidor”. Por otro lado, en Juran [10] exponen a la calidad como “Es la adecuación para el uso satisfaciendo las necesidades del cliente”. Cuatrecasas [11] señala que “Calidad es el conjunto de características que posee un producto o servicio obtenidos en un sistema productivo, así como su capacidad de satisfacción de los requisitos

del usuario”. Y por último la norma ISO 9000:2005 [12], plantea que la calidad es el grado en el que un conjunto de características inherentes que cumplen con los requisitos.

Desde distintos puntos de vista de diferentes autores, se ha abordado el concepto de calidad, pero se observan dos comunes denominadores: la satisfacción del cliente y el cumplimiento de los requisitos. Por lo tanto, se puede entender que la calidad va enfocada al cliente, donde cumplir con las expectativas del destinatario final (cliente) es el objetivo principal de concepto.

Es importante considerar que la calidad debería tomarse como una filosofía de trabajo para una organización. Prestar un servicio de calidad involucra una actividad pro-activa que incorpora el control, el aseguramiento, el perfeccionamiento y la planificación de un conjunto de actividades de carácter administrativo, dirigidas a los determinados niveles de calidad por parte de la organización y así el mantenimiento de esos niveles adquiridos [13].

El interés de la sociedad por la calidad es tan antiguo como el origen de las sociedades, por lo que tanto el concepto, como las formas de gestionar la calidad han ido evolucionando progresivamente.

La inspección es la primera etapa en la búsqueda de la calidad. Para algunos autores se inicia en la fábrica de Ford en 1910, la cual utilizaba equipos de inspección para comparar los productos de su cadena de producción con los estándares establecidos para el proyecto [14].

Durante esta etapa, se consideró que la inspección era la única manera de asegurar la calidad, reflejándose esto en el pensamiento y literatura técnica de la época. La ejecución de la práctica se orientó a tareas tales como la selección y clasificación de los productos, el rescate de productos de lotes dañados, reprocesamiento, la ejecución de mezclas para salvar la materia prima con daños leves, la toma de acciones correctivas y la búsqueda de las fuentes de no conformidad [1].

El desarrollo de la producción en masa, la especialización, el incremento en la complejidad de los procesos de producción y la introducción de la economía de mercado centrada en la competencia

y en la necesidad de reducir los precios, hecho que implica reducir costes de mercadería y de proceso, determinó la puesta en marcha de métodos para mejorar la eficiencia de las líneas de producción. A partir de esta base, se hace evidente la evolución de las empresas a una nueva etapa de la calidad, que comienza con la introducción de la filosofía y prácticas del Control de Calidad [14, 1].

El Control de Calidad hace referencia a técnicas y actividades de carácter operacional. Se orienta a mantener bajo control los procesos y eliminar las causas que generan comportamientos insatisfactorios en etapas importantes del ciclo de calidad, para conseguir mejores resultados económicos (ISO 8402) en [1].

En esta etapa, existe ya un método de calidad, siendo la inspección parte del Control de Calidad. La filosofía y la práctica del Control de Calidad se orientan al desarrollo de manuales de calidad, la recolección de información sobre el comportamiento de los procesos, utilización de la estadística básica del control de calidad y ejecución de autocontrol. Además del análisis y ensayo de materias primas, de productos en proceso y productos terminados. Y por último, se establecen los procedimientos para la elaboración, control y difusión de informes [1].

A partir de los años 60, se inició en EEUU el movimiento de protección de los consumidores y la necesidad de asegurar que los productos que eran presentados en el mercado cumplieran, entre otros, altos estándares de seguridad conforme con el uso que el cliente iba a dar a ese producto. No obstante, el Control de Calidad no le garantizaba al consumidor el cumplimiento de sus demandas cambiantes y tampoco los resultados económicos de la gestión empresarial, por lo tanto muchas empresas innovan en el campo de la calidad dando paso a una nueva etapa con la introducción de la filosofía y prácticas del Aseguramiento de la Calidad [14, 1].

En este periodo se reconoció que la calidad podía quedar garantizada en el lugar de la fabricación mediante el establecimiento de un sistema de calidad, que permite satisfacer las necesidades del cliente final. Esta garantía podría ser llevada a cabo mediante el desarrollo de un sistema interno que, con el tiempo, genere datos, que nos señale que el producto ha sido fabricado según las especificaciones y que

cualquier error había sido detectado y eliminado del sistema [14].

En la etapa de Aseguramiento de la Calidad, se aplicó el concepto de la calidad en todas las etapas del ciclo del producto dentro de la organización: diseño del producto, diseño de procesos, producción, ventas y servicios posventa. En cada una de las etapas se aplicaron un conjunto de técnicas englobadas, muchas de ellas bajo el nombre de ingeniería de la calidad [14].

A pesar de los grandes avances que impulsó el Aseguramiento de la Calidad, no fue suficiente para garantizar al consumidos, el cumplimiento de sus demandas y tampoco se obtuvieron los resultados económicos deseados en las empresas. Es por ello que frente a las nuevas necesidades, las empresas tienen que volver a cambiar, dando inicio a una nueva etapa que se caracteriza por la introducción de nuevas teorías y prácticas esta nueva etapa, es la Gestión de la Calidad Total.

La Gestión de la Calidad Total o TQM (Total Quality Management). Es una práctica gerencial para el mejoramiento continuo de los resultados en cada área de actividad de la empresa y en cada uno de los niveles funcionales, utilizando todos los recursos disponibles y al menor costo. El proceso de mejoramiento se orienta hacia la satisfacción completa del consumidor, considerándose el recurso humano como uno más de la organización [14].

En esta nueva evolución, en el concepto filosófico de la calidad se introduce a lo ya existente (inspección, control de calidad y aseguramiento de la calidad), la participación del proveedor y del consumidor como socios estratégicos de la empresa. La filosofía y el enfoque es satisfacer el 100% de las veces de la demandas, tanto del consumidor interno como externo [14].

En la década de los 90, las principales corporaciones reconocieron que cada año se desperdiciaban miles de millones de dólares en software que no tenían las características ni la funcionalidad que se había prometido [8]. Si bien la industria del software es nueva, ha tenido que madurar rápidamente, tal como lo exigen los avances tecnológicos y su alta participación al interior de las empresas. En la industria del software se pueden evidenciar

necesidades de satisfacción del cliente de productos o servicios de software, de reducción de recursos invertidos en proyectos de software y de la efectiva asignación de recursos humanos [15].

Generalmente, la calidad de software es vista como un proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan [8]. Y por otro lado, la IEEE Standards Association (IEEE-SA), la define como el grado con que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario [16].

Según Pressman [8] la calidad de software puede ser estandarizada a través de tres puntos importantes: un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad. Un producto útil entrega contenido, funciones y características que el usuario final desea. Al agregar valor para el producto y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales.

A través de estos tres puntos de estandarización, se puede visualizar la calidad en un producto de software. Pero, ¿cómo se logra la calidad?, ¿por qué medio se puede obtener calidad en el software? Para Pressman [8], un software de alta calidad se logra a través de una buena administración del proyecto y de una correcta práctica de ingeniería de software. Esto se puede obtener mediante la aplicación de cuatro aspectos. El primero es Métodos de la ingeniería de software, si se quiere crear un software de alta calidad, se necesario comprender la problemática, construir un diseño que esté acorde al problema que se quiere resolver y que al mismo tiempo tenga características que lleven al software a ser un producto de calidad. El segundo aspecto son las Técnicas de administración de proyectos, es frecuente que la calidad del software reciba influencias tanto de las decisiones administrativas como de las tecnologías, incluso las mejores prácticas de la ingeniería de software pueden ser arruinadas por malas decisiones gerenciales y por acciones cuestionables de la administración del proyecto. El tercer aspecto es el Control de Calidad, que incluye un conjunto de actividades

y tareas de ingeniería de software que ayudan a lograr cumplir con las metas en el transcurso del proyecto. Y, finalmente el último aspecto es el Aseguramiento de la Calidad, que a través de herramientas, procesos, auditorías y métodos entrega un apoyo sólido a la administración del proyecto, para elaborar un software de alta calidad.

Aseguramiento de la Calidad del Software

El aseguramiento de la calidad (ACS), es una actividad que se aplica a todo el proceso del software. El ACS incluye procedimientos para la aplicación eficaz de métodos y herramientas, supervisa las actividades de control de calidad, tales como revisiones técnicas y las pruebas del software, procedimientos para la administración de cambio y elaboración de reportes [8]. Para llevar a cabo el aseguramiento de la calidad del software de manera adecuada, deben recabarse, evaluarse y divulgarse datos sobre el proceso de ingeniería de software. Los métodos estadísticos aplicados al ACS ayudan a mejorar la calidad del producto y del proceso de software mismo [8]. Según Pressman, uno de los principios fundamentales para lograr la calidad de un producto de software es el Aseguramiento de Calidad, es desde ahí que nace esta propuesta metodológica de Aseguramiento a Calidad, a través de un enfoque práctico.

TRABAJOS RELACIONADOS

Para entender mejor el contexto nos enfocaremos en la descripción de diferentes trabajos y documentos que están directamente relacionados con un método de aseguramiento de la calidad para el desarrollo de software, de esta manera, se intenta recabar información de diferentes procesos que de una u otra manera intentan asegurar la calidad en el transcurso del desarrollo de un sistema.

El trabajo presentado por Diez [2] presenta un conjunto de acciones y métodos de ACS. El modelo general de ACS, se documenta en un plan general de aseguramiento de la calidad, flexibilidad que a través de diferentes fases y módulos ayuda a desarrollar un producto de calidad, permitiendo la adaptación del modelo genérico o plan general de aseguramiento de la calidad del software, a todo tipo de proyectos. Se formaliza a través de los planes específicos de aseguramiento de la calidad del software para cada proyecto. Este enfoque no presenta ser exclusivo y

en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podría ser su complemento, adaptándolo convenientemente. Cabe destacar, que esta propuesta identifica un equipo especializado en el aseguramiento de calidad dentro de la empresa, encargado de realizar todas las tareas de SQA.

Por una parte, el trabajo de Aguirre [17], a pesar de que no es un método o plan de aseguramiento de la calidad, su objetivo es el mismo: satisfacer los requerimientos del usuario a través de una calidad esperada. Este marco metodológico, está basado en diferentes estándares de calidad, que a su vez le proporcionar una serie de características para poder guiar un desarrollo de software de forma exitosa. Cabe destacar que este marco está desarrollado pensando en lo complicado que es para las pymes aplicar métodos o estándares de calidad, debido principalmente a aspectos relacionados con su complejidad, costos, restricciones etc. Es por ello, que se crea este método de V&V para beneficiar a las empresas desarrolladoras de software en Perú, el desarrollo de sus proyectos de software.

Por otra parte, el trabajo de Paladines [18], describe un plan de aseguramiento de la calidad a través de diferentes actividades de verificación con checklist, de las diferentes etapas del desarrollo del proyecto, que no hace distinción entre el método de desarrollo y la estrategia de aseguramiento de la calidad. Este plan de gestión de la calidad, solo se encarga de controlar si se cumplen con los requisitos al final de cada etapa del mismo y no es una guía de aseguramiento de la calidad para el desarrollo del proyecto, por lo tanto solo es un control de calidad.

Y por último, en Gómez [3] se presenta una revisión sistemática de la literatura acerca del tópico: adaptaciones en la mejora del proceso software (SPI) en las MiPyMEs, en el período comprendido de 1995 a diciembre de 2013. Su objetivo es, presentar información actualizada sobre las tendencias de este tópico como son países y sectores que abordan el tema, así como modelos, metodologías, estándares, y procesos de soporte del área de calidad reportados en este tipo de empresas. Esta revisión sistemática se centra en aportaciones reportadas sobre procesos de soporte del área de calidad del ciclo de vida del software.

De este trabajo se puede extraer información valiosa sobre el contexto de aseguramiento de la calidad, como por ejemplo: El modelo más usado es CMMi con un 40%, seguido por SPICE con 21% y por CMM con un 17% [3]. Estos resultados muestran que los modelos de calidad tratados en los estudios revisados, son adaptaciones que se basan principalmente en los modelos para las grandes empresas. Aun cuando existen otros modelos de calidad para las pequeñas empresas. Esto sugiere cierto grado de inmadurez en el conocimiento, la aceptación y la experimentación de dichos modelos en las pequeñas empresas [3]. Además, los procesos de soporte del área de calidad del software que reciben contribuciones de los estudios seleccionados son: aseguramiento de la calidad con un 46%, verificación con un 41%, validación con 11%, y auditoria con 2%. Estos resultados sugieren que la comunidad de Ingeniería de Software está priorizando la formalización y la verificación de los procesos que ayuden a asegurar la calidad del software. Algunas razones que quizás está potenciando la investigación en dichos procesos son las especificaciones incompletas, cambios continuos en requerimientos, falta de formalidad en procesos y metodologías, así como la carencia de control en los procesos problemas comunes en las pequeñas empresas desarrolladoras de software [3].

Se pueden identificar, algunas falencias presentadas por los trabajos anteriores, tales como: la incorporación de un equipo o personal a cargo del ACS que encare no solo el proyecto, sino también los gastos de la empresa; los procesos de V&V son solamente para las etapas de implantación y testing, dejando de lado las fases iniciales y de desarrollo de un producto de software; y, por último, un plan de aseguramiento de la calidad se encarga de gestionar la calidad en todo el desarrollo del proyecto, por lo tanto no es una medida de control.

A pesar de ello, la búsqueda de diferentes autores, por proponer métodos, actividades, etc. que aseguren de una u otra forma la calidad en el desarrollo de software [2][17][18], las dificultades que tiene las pequeñas empresas desarrolladores de software [4][5] y los resultados de [3], dan cuenta de la necesidad de métodos o actividades que aseguren la calidad en el desarrollo de productos de software, para pequeñas y medianas empresas.

Es por ello, que en este trabajo se propone un método de aseguramiento de la calidad, para cualquier

empresa o proyecto de desarrollo de software, que no encarezca los costos, sea transversal y gestione la calidad en todas las fases o etapas de una metodología de desarrollo.

PROPUESTA DE ASEGURAMIENTO DE LA CALIDAD

El método de ACS que se propone en este trabajo, permitirá a cualquier proyecto de desarrollo de software, proporcionar calidad al producto final. Este enfoque práctico está diseñado, para adaptarse o acoplarse a cualquier Metodología de Desarrollo y de esta manera administrar la calidad en desarrollo de un proyecto. Esta propuesta está basada en los conceptos y referencias que según Pressman [8] deberían incluir un Aseguramiento de la Calidad del software: 1) un proceso de ACS, 2) tareas específicas de aseguramiento de la calidad y control de la calidad, 3) prácticas eficientes de ingeniería de software (métodos y herramientas), 4) control de todos los productos del trabajo de software y de los cambios que sufren, 5) un procedimiento para garantizar el cumplimiento de los estándares del desarrollo de software, y 6) mecanismos de medición y reporte.

Por lo tanto, tomado en consideración lo que plantea Pressman [8], se desarrolla y propone un método de ACS. Esta propuesta está compuesta por Esencia y

Herramientas el que será medido por Métricas, que buscan no solo una forma de trabajar y administrar la calidad, sino también la posibilidad de mejorar los procesos de desarrollo a través de la recolección y análisis de errores.

Método de ACS

Como se mencionó anteriormente y puede verse en la Figura 2, esta Propuesta Metodológica de Aseguramiento de la Calidad, se puede adaptar o acoplar a cualquier Metodología de Desarrollo que se esté utilizando, sin tener la necesidad de agregar un miembro más o un equipo paralelo de ACS.

A continuación pasaremos a describir cada una de las tres actividades o tareas que componen la metodología. Es importante destacar que estos tres componentes se relacionan entre sí, entendiendo que una de las herramientas, puede ser una forma de medir o que a su vez, una herramienta tenga el concepto o la filosofía PDCA.

Esencia

Esta actividad en el método de ACS, hace referencia a la manera de cómo se debe comprender, por el grupo de trabajo, la forma de trabajar, en base a un objetivo que es proporcionar calidad al producto de software. Si todo el equipo de trabajo, desde el líder de equipo, pasando por todos los roles del grupo, trabajaran entendiendo que la calidad es

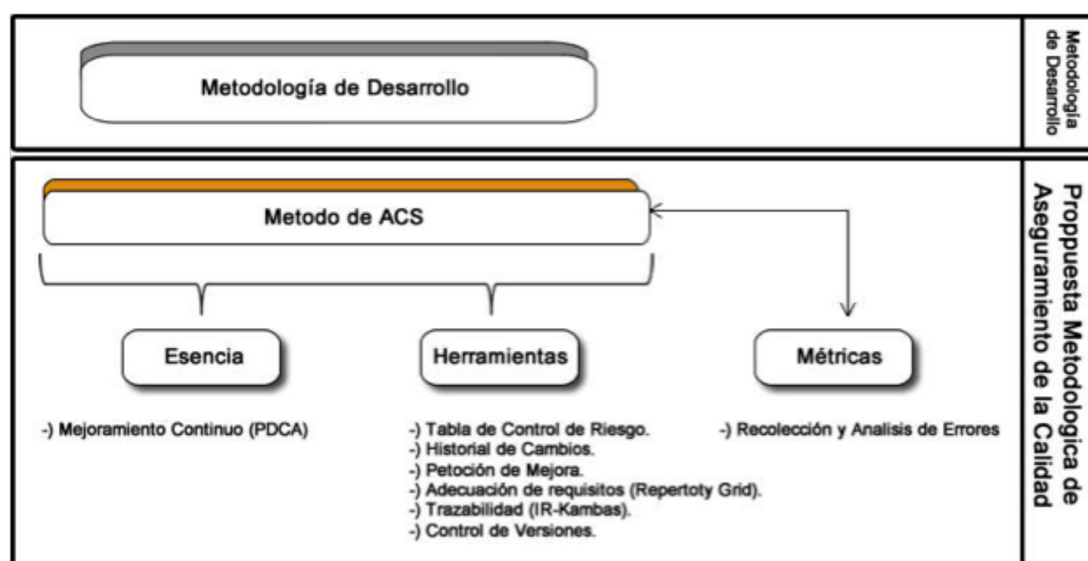


Figura 2. Propuesta metodológica de ACS.

fundamental en el desarrollo de un producto y su objetivo es satisfacer las necesidades del cliente, la administración de la calidad no sería un problema, ya que todos estarían trabajando en función de ese objetivo.

Mejoramiento Continuo

La filosofía de mejora continua es una herramienta ideal, para que el equipo de trabajo entienda y busque la calidad en el desarrollo del producto. El proceso de mejora continua es una constante refinación de las normas para responder de una forma dinámica a las exigencias del cliente y las oportunidades de mejora de los procesos. Para que esto suceda, la administración debe establecer primero el estándar o la base en los procesos o actividades, para que, posteriormente, el ciclo PDCA desempeñe su función reguladora [20], mejorando los estándares establecidos. De esta manera, el ciclo PDCA permite el aprendizaje organizacional y el logro de mejores estándares [9].

Plan (Planificar o Planear), énfasis a la planificación del proyecto.

Do (Hacer), corresponde a realizar, fabricar o trabajar el producto planificado.

Check (Revisar o Comprobar), para confirmar si el cliente está satisfecho o el proceso está según especificaciones.

Act (Actuar), si se presenta algún reclamo, se actúa sobre el problema sin tener que esperar que finalice el proceso, luego se vuelve a la fase de

planificación, volviendo al ciclo PDCA, para obtener un mejoramiento.

Herramientas

Las herramientas, en la propuesta metodológica de ACS, son un conjunto de técnicas y artefactos que ayudarán a mantener el control y a la vez administrar la calidad en el desarrollo desde el inicio hasta el final, de un producto de software. A continuación se describirán y mostrarán las herramientas que propone este método de ACS.

Tabla de Control de Riesgo

La Figura 1 muestra la tabla de Control de Riesgo, que tiene como objetivo identificar, controlar y eliminar las fuentes de riesgo antes de que empiecen a afectar el cumplimiento de los objetivos del proyecto. De esta manera se pueden evaluar y estimar el impacto de los riesgos posibles y a su vez establecer un plan de contingencia para mitigarlos en el caso de que el problema se presente, teniendo como objetivo la pro actividad. Por lo tanto: 1) se inicia antes del trabajo técnico, 2) se categoriza o clasifica según su impacto, 3) identifica los riesgos potenciales, valorando su probabilidad de impacto y 4) establecer un plan.

Historial de Cambio

La Figura 4 muestra el Documento Historial de Cambio, que permite al equipo de trabajo llevar un registro de los cambios o mejoras solicitadas en el Documento de Petición de Mejora, junto con ello también admite llevar un control de los documentos o entregables, tanto sus versiones como alguna modificación en ellos.

Riesgo	Categoría	Probabilidad	Impacto	Plan de Acción

Clasificación en Categoría y Factores de Riesgo el nivel de Impacto.

Componentes de Riesgo	Rendimiento	Factores de Riesgo	Despreciable
	Coste		Marginal
	Mantenibilidad		Critico
	Planificación		Catastrófica

Figura 3. Tabla de Control de Riesgo.

ENTREGABLE/ PETICIÓN DE CAMBIO/ OTROS	VERSION /N° PETICION	FECHA	PRIORIDAD	DESCRIPCIÓN

Figura 4. Historial de Cambio.

Documento de Petición de Mejora

Este documento llamado Petición de Mejora, que se puede observar en la Figura 5, tiene como objetivo priorizar y mostrar las modificaciones que proponen o recomiendan los Stakeholders o Clientes. De

esta manera, se podrán analizar dichos cambios o propuestas e implementarlas en el proyecto, controlando de modo ordenado los cambios en el desarrollo del sistema. Cabe destacar que este documento sirve para gestionar cambios o mejoras

Petición de Mejora	
Solicitante:	Número de Petición:
Fecha:	Nombre del Proyecto:
Prioridad de Petición:	
Prioridad: A) Alta/Esencial B) Media/Deseada C) Baja/Opcional	
ANTES	DESPUÉS
<div>Firma del Solicitante</div>	

Figura 5. Documento de Petición de Mejoras.

ya sea en los requisitos del sistema, entregables, artefactos o documentación del proyecto, luego de ser testeada o implementado.

Adecuación de requisitos (Repertory Grid)

Uno de los problemas fundamentales en el desarrollo de software es la mala calidad de la educación de los requisitos del sistema, provocando costos importantes en los proyectos. Por lo tanto, para evitar los problemas anteriores y desarrollar productos que satisfagan al cliente, se propone utilizar el emparrillado (Repertory Grid) planteada en [21]. La Tabla 1 muestra un ejemplo de Repertory Grid, la cual permite determinar el valor de adecuación de dichos requisitos a las necesidades del usuario, a mayor adecuación, mayor calidad.

Tal como lo plantea Meléndez [21], por un lado el polo nominal o de semejanza de los constructos dicotómicos a utilizar, se estructura de la siguiente forma:

- **Correcto**, si, y solo si, cada requisito declarado se encuentra en el producto.
- **Inequívoco**, si, o solo si, cada requisito declarado tiene una sola interpretación. Debe ser inequívoco para aquellos que lo crean y para aquellos que lo usen.
- **Completo**, si, y solo si, están relacionados con la funcionalidad, el desarrollo, las restricciones de diseño, los atributos y las interfaces externas.
- **Consistente**, si, y solo si, ningún subconjunto de requisitos individuales generó conflictos en él.
- **Importante**, si, y solo si, él tiene un identificador para indicar la importancia que lo relaciona al producto.
- **Estable**, si, y solo si, el número de cambios realizados, o que se espera realizar, es mínimo.
- **Comprobable**, si, y solo si, existe algún proceso rentable finito con que una persona o la máquina pueda verificar que el producto reúne el requisito.
- **Modificable**, si, y solo si, su estructura y estilo son tales que pueden hacer cualquier cambio fácilmente, completamente y de forma consistente conservando la estructura y estilo.
- **Identificable**, si, y solo si, su origen está claro y facilita su referencia en el desarrollo futuro o documentación del mismo, debiendo ser tanto identificable dirigido hacia tras, como identificable dirigido hacia a delante.

Por el otro lado, el polo de contraste está compuesto por, Incorrecto, Ambiguo, Incompleto, Débil, Intrascendente, Inestable, No Comprobable, No Cambiable, No Reconocible.

Al igual que en la propuesta de Meléndez [21], los constructos a utilizar serán del tipo multivaluados (1-9) en donde 9 es la máxima aproximación al polo nominal y 1 su homólogo al polo de contraste. De esta manera, se obtendrán los niveles de adecuación que posee cada una de los requisitos elicitados: Verde (7-9), Naranja (4-6), y rojo (1-3).

Trazabilidad (Codificación IR-Kanban)

El no tener un control de trazabilidad, asociado a identificar los diferentes documentos o artefactos en un desarrollo de software, podrían causar problemas y por lo tanto perjudicar en la calidad del producto.

Tabla 1. Ejemplo de Aplicación de Repertory Grid.

		Requisitos funcionales			
		R1	R2	R3	
CONSTRUCTOR	Correcto	3	9	9	Incorrecto
	Inequívoco	9	9	6	Ambiguo
	Completo	9	2	9	Incompleto
	Consistente	7	9	9	Débil
	Importante	9	4	9	Intrascendente
	Estable	9	8	9	Inestable
	Verificable	1	9	9	No Comprobable
	Modificable	9	9	9	No Cambiable
	Identificable	9	1	9	No reconocible
	Trazable	5	9	2	No Trazable

Para esto se propone a IR-Kanban [22], que es un sistema de codificación del cual se puede obtener un correcto control de la documentación asociada al proyecto como: SRS, Diagramas de flujos, casos de uso, entre otros. Ayudando a identificar diferentes aspectos a través de los sub códigos que lo componen. En la Figura 6, se observa la estructura de IR-Kanban donde el código resultante, estará conformado por segmentos de información, agrupados de forma similar a las MAC.

$$P_2 P_1 P_0 - H_2 H_1 H_0 - I_2 I_1 I_0 - C_2 C_1 C_0 - V_1 V_0 - E_2 E_1 E_0 - B_2 B_1 B_0 - F_0 - S_2 S_1 S_0 - O_0$$

Figura 6. Estructura del Sistema de Codificación.

La Tabla 2, describe la estructura del sistema IR-Kanbas.

Tabla 2. Descripción del Sistema de Codificación IR-Kanban.

Sub Código	Descripción
P ₂ P ₁ P ₀	Código Identificador de proyecto, por ejemplo "P15", haciendo referencia al proyecto N° 15.
H ₂ H ₁ H ₀	Código de 3 dígitos que identifica la naturaleza del artículo que esta siendo referenciado, el cual puede ser: Documento de Especificación de Requisitos (SRS), Caso de Uso (CUF).
I ₂ I ₁ I ₀	Dígitos para identificar la pertenencia del artículo con algún otro de mayor nivel, por ejemplo si el artículo es identificado como un caso de uso (CUF), esta opción identifica el paquete al cual pertenece.
C ₂ C ₁ C ₀	Código de 3 dígitos que sirve para identificar el artículo, el cual se incrementa correlativamente.
V ₁ V ₀	Hace referencia a la versión del artículo, por ejemplo "12", haciendo referencia a la 12ª interacción o versión.
E ₂ E ₁ E ₀	Código de 3 dígitos que identifica el equipo de trabajo "E03", referenciando al equipo N° 3.
B ₂ B ₁ B ₀	Identificador del jefe de proyecto o la persona encargada de la validación interna de el artículo.
F ₀	Dígito de Flag que representa el estado de validación interna del artículo, por parte del jefe de proyecto o la persona encargada para ello. Este indicador puede ser: En espera (0), Validado (1), Rechazado (2), Incompleto (3), entre ellos.
S ₂ S ₁ S ₀	Identificador del stakeholders o la persona encargada de la validación externa del artículo (aprobación).
O ₀	Dígito que representa el estado de validación externa del artículo (aprobación), por parte del stakeholders o la persona encargada para ello. Este indicador puede ser: En espera (0), Aprobado (1), Rechazado (2), Incompleto (3), entre otros.

Control de Versiones

El código fuente es el elemento primordial en un desarrollo de un producto de software y poder administrarlo de forma eficiente, siendo fundamental

a la hora de proporcionar calidad al proyecto de software. Una de las herramientas propuestas del método ACS, es el sistema de control de versiones (SCV). Los SCV son una herramienta esencial para manejar proyectos de software. Proporcionan una serie de funcionalidades claves para el desarrollo de proyectos como es el control de cambios en el código, la reversibilidad de dichos cambios, y la posibilidad de colaborar en el desarrollo del código [15].

Los principales beneficios de esta herramienta según O'Sullivan [23] en [24] son: 1) cualquier revisión almacenada de un archivo puede ser recuperada para visualizarse o modificarse, 2) pueden desplegarse las diferencias entre distintas versiones, 3) las correcciones pueden ser creadas automáticamente, 4) múltiples desarrolladores pueden trabajar simultáneamente en el mismo proyecto o archivo sin pérdida de datos, 5) los proyectos pueden ser divididos para permitir el desarrollo simultáneo en varias versiones (estas divisiones pueden ser fusionadas para alcanzar el objetivo principal del desarrollo) y 6) el desarrollo distribuido, es soportado a través de las redes de datos con diferentes mecanismos de autenticación. Uno de los más utilizados es Git, inicialmente desarrollado para Linux. Git permite a varios programadores trabajar paralelamente con sus propias copias de trabajo obtenidas de un repositorio, como lo efectúan todos los SCV distribuidos [24].

Métricas

La única manera de mejorar es medir como se está haciendo algo [8]. Para ello, este modelo propone una herramienta de ACS estadístico que reúne y clasifica errores para luego analizarlos y poder mejorar sustancial a la calidad, en el desarrollo de proyectos futuros.

Según Pressman los errores se pueden clasificar en las siguientes causas:

- Especificaciones erróneas o incompletas (EEI).
- Mala interpretación de la comunicación con el cliente (MCC).
- Desviación intencional de las especificaciones (DIE).
- Violación de los estándares de programación (VEP).
- Errores en la representación de los datos (ERD).
- Interfaz componente inconsistente (ICI).

- Error en el diseño lógico (EDL).
- Prueba incompleta o errónea (PIE).
- Documentación inexacta o incompleta (DII).
- Errores en la traducción del lenguaje de programación del diseño (LPD).
- Interfaz humano/computadora ambigua o inconsistente (IHC).
- Varios (V).

Para aplicar el ACS estadísticos se elabora una tabla, como se observa en la Tabla 3, que contiene todas las posibles causas de errores, clasificadas en: Serio, Moderado y Menor. De esta manera se puede observar e identificar los errores más críticos o vitales, se puede dar comienzo a sus acciones correctivas.

Tabla 3. Colección de Datos para ACS Estadístico.

	Total		Serio		Moderado		Menor	
Error	Nº	%	Nº	%	Nº	%	Nº	%
EII	186	45	48	50	73	41	65	46
MCC	123	30	13	13	86	49	24	17
PIE	55	13	18	18	5	2	32	23
DII	21	5	14	14	5	2	2	1
LDP	25	6	3	3	6	3	16	11
Total	410	100	96	100	100	100	139	100

Es importante destacar que acciones correctivas se centran sobre las pocas causas vitales. En tanto estas se corrigen, nuevas candidatas se van a la cumbre de la pila.

ESTUDIO DE CASO

Para ilustrar el uso y aplicación de la propuesta metodológica de Aseguramiento de la Calidad, se presenta el estudio de caso del proyecto STCplaza. El proyecto Sistema Transaccional de Comercial Plaza S.A, se traduce en la creación de una Aplicación Web que soporta las tres áreas de negocio de la empresa: Compra y Venta, Taller Mecánico y Taller de Mantención. Para lograr apoyar y soportar el modelo de negocio, la aplicación consta de cuatro módulos: Administración, Datos de Empresa, Bodega y Taller, más un Inicio del Sistema. Este proyecto, fue creado a través de una propuesta metodológica de desarrollo llamada LEAN-RUP.

LEAN-RUP divide el proceso en cuatro fases (Inicio, Elaboración, Construcción y Transición), dentro de las cuales se realizan varias iteraciones según el proyecto y en las que se hace un mayor o menos esfuerzo en las distintas actividades. Además posee distintos flujos de trabajo (Modelo de negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Gestión y Configuración de Cambios, Gestión de Proyecto y Entorno), que son un conjunto de actividades ordenadas para lograr un resultado. Estas actividades son representadas a través de un conjunto de artefactos, relacionados con las fases dentro de todo el proyecto.

En los flujos de trabajo Gestión y Configuración de Cambios y Gestión de Proyecto, es donde se inserta el modelo de ACS, que tiene como objetivo administrar la calidad en el desarrollo de este proyecto. Estos dos flujos de trabajo, son transversales en todas las fases de la metodología LEAN-RUP.

A continuación, se pasará a describir las herramientas y la forma como se gestionó la calidad en el proyecto. Cabe destacar que no toda la propuesta de ACS se utilizó en el desarrollo de este trabajo. Debido a las características de este proyecto, como por ejemplo, el sistema fue desarrollado por un solo programador, por lo tanto el control de versiones no fue necesario.

La primera herramienta que se utilizó fue la Tabla de Control de Riesgos. La Figura 7, muestra la utilización de esta herramienta en el proyecto STCplaza. La que fue llenada con algunos riesgos importantes para el grupo de trabajo. Como por ejemplo, uno de los principales riesgos detectados fue:

Riesgo: Falta de experiencia en las herramientas de desarrollo.

Categoría: Entorno de Desarrollo.

Probabilidad: 80%.

Impacto: Planificación, Crítico.

Plan de Acción: Se estudian las herramientas a utilizar. Se crean pequeños códigos relacionados al producto.

Para la trazabilidad, se utilizó IR-Kanbas para cada artefacto. La Figura 8 muestra un ejemplo de la

Código: P01-SRS-FI-02-1.1-RL04-RL01-1-ST01-1

Figura 8. Ejemplo de Trazabilidad con IR-Kanbas.

trazabilidad del Documento de Especificación de Requisitos de Software.

Este código, consiste en mantener la trazabilidad del documento de Especificación de Requisitos, el cual se pasara a describir a continuación:

- P01: Corresponde al proyecto STCplaza, denominado P01.
- SRS: Corresponde al nombre del Documento de Especificación de Requisitos de Software (SRS).
- FI: Corresponde a un Documento de la Fase de Inicio (FI), del método de desarrollo LEAN-RUP.

Riesgo	Categoría	Probabilidad	Impacto	Plan de Acción
1. Dificultad de migración de los datos antiguos	Entrenamiento de usuarios	30%	- Planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
2. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
3. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
4. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
5. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
6. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
7. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
8. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
9. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.
10. Falta de control de cambios controla. Monte los requerimientos	Proyecto	30%	- Control y planificación - Control	- Se asignan los recursos antes de migrar. Se realizan pruebas piloto. Se asignan recursos para el control de la migración.

Figura 1. Tabla de Riesgo del Proyecto STCplaza.


Petición de Mejora	
Solicitante: MARCELO A.	Numero de Petición: 01
Fecha: 20/09	Nombre del Proyecto: Sistema Informatizado Comercial Plaza (STCplaza)
Prioridad de Petición	
Prioridad: A) Alta/Esencial	B) Media/Deseada C) Baja/Opcional
Antes	Después
- FASE DE LECTURA - COPIAS EMPRESA - Definir que la orden de compra a por proveedor. Ref: RF 01 - De Orden de Compra a factura de venta Ref: RF 01.	- Se modifica el formato de la impresión de la orden de compra al formato actual de la empresa. - Se cambian los colores del sistema por los institucionales, además se agrega el logo de la empresa en cada módulo. - Ref: RF 01 se modifica donde se indica que la orden de compra a por proveedor. - Ref: RF 01: se cambia que el Ingreso sea de Orden de Compra - factura de venta.
 Firma del Solicitante:	

Figura 8. Documento de Petición de Mejora.

- 02: Corresponde al Segundo Documento que se entrega en esa Fase (02).
- 1.1: Corresponde a la Versión (1.1) del Documento.
- RL04: Corresponde al Rol de quien lo creo, en este caso RL04, corresponde al Ingeniero de Software.
- RL01: Corresponde al Rol der Jefe de proyecto (RL01).
- 1: Corresponde, al estado de validación, 1 (validado) por el Jefe de Proyecto.
- ST01: Corresponde al Stakeholder del proyecto (ST01).
- 1: Corresponde, al estado de Validación, 1 (validado) por el Stakeholder.

Para poder controlar los cambios en el proyecto, se utilizó el Documento de Petición de Mejora, el cual fue llenado por el Stakeholder con los cambios solicitados a medida que fue avanzando el proyecto. Este documento se utilizó hasta el término del trabajo. Para llevar un registro de todos los cambios o modificaciones que se realizaron en torno al proyecto, desde las peticiones de mejora, cambios en los requisitos, modificaciones provenientes desde el equipo de trabajo, etc. se utilizó el Historial de Cambios. Este documento era de total conocimiento de todos los interesados en el desarrollo. En la Figura 8, se observa el Documento de Petición de Mejora utilizado, para gestionar los cambios en el proyecto STCplaza.

Para la toma de requisitos, se utilizó un formato de ordenamiento de los requisitos, siendo este no un

elemento de ACS. Lo que propone el método, es la forma de cómo fueron adecuando estos requisitos a través de la herramienta Repertory Grid, para así de esta manera poder obtener requisitos de alta calidad, lo más parecido a las necesidades del cliente. En la Figura 9 se observa el formato de ordenamiento del requisito Genera Sobre Impresión, que además tiene la propuesta de adecuación de requisitos del modelo de ACS la Repertory Grid.

En la Tabla 4, se puede observar los resultados obtenidos al ponderar los valores obtenidos en la primera educación de requisitos. Para este proyecto se realizaron dos adecuaciones de requisitos, obteniendo un alto nivel de ajuste en la segunda adecuación.

Y por último, la comprensión del equipo de trabajo sobre lo importante que es poder proporcionar calidad en el desarrollo del proyecto, llevó que al término de cada iteración, se realizaran reuniones en las cuales se corregían los errores detectados hasta ese momento y se planeaban los pasos a seguir, teniendo como fin entregar un producto que satisficiera las necesidades del cliente.

La utilización de métricas no se ve refleja en este caso de estudio, debido a que es un único proyecto de software. Para la aplicación del ACS estadístico, es necesario la entrega de varios proyectos de desarrollo, los que permitirán una mayor recolección de datos y a su vez, mayor información sobre los posibles focos de mejora en procesos, tareas o actividades en el desarrollo de un producto de software.

Tabla 4. Primera adecuación de Requisitos.

[illegible]

CONCLUSIONES

El éxito de un producto de software está, básicamente reflejado en la satisfacción de cliente. Esto quiere decir, entregar un producto en el tiempo estimado y que no sobrepase los costos cumpliendo con los requisitos declarados, para lograr aquello. Es necesario tener un plan que trace la ruta para la creación del producto y a su vez tener una estrategia que administre y controle la calidad en el desarrollo de un producto de software. Para esto, se propuso un método de aseguramiento de la calidad, como estrategia y actividad paraguas en el desarrollo de un proyecto de software, que a través de sus diferentes actividades, tareas y conceptos, que se pueden acoplar en la metodología de desarrollo o en diferentes etapas del desarrollo del proyecto. Y que no solo administre la calidad, sino también ayude a mejorar el trabajo del equipo de desarrollo.

Esta propuesta de ACS, fue puesta en práctica en el desarrollo de una aplicación web para la empresa Comercial Plaza S.A. Este modelo de ACS, administró de forma exitosa la calidad en el desarrollo del proyecto. Por una parte, haciendo trabajar al equipo, comprometidos desde un comienzo del proyecto con la calidad, no solo externa, desarrollando un sistema que los stakeholders percibieran desde un inicio que se estaba cumpliendo sus expectativas, sino también de forma interna, para que el trabajo fuera coordinado, sin tener mayores complicaciones.

Por otra parte, la utilización de herramientas que permitieran el control y la administración del proyecto favoreció que el trabajo se mantuviese entre lo planeado, ayudando y aportando a la metodología de desarrollo no solo la gestión de la calidad, sino también al desarrollo de la mismas actividades pertenecientes a la metodología, como por ejemplo. Las diferentes herramientas que posee este método de ACS, permite la flexibilidad y adaptación a los cambios. Beneficiando de mejor manera una característica principal de la metodología de desarrollo utilizada en este proyecto. Todo esto ayudó a terminar un proyecto en los plazos que se fijaron y con las características que el cliente necesitaba.

Cabe destacar que, como se mencionó anteriormente, las métricas son a plazos más largos, donde el volumen de información a recabar, sea determinante para tomar decisiones metodológica de aseguramiento

de la calidad, no ha sido probada en equipos de más de tres integrantes. En el futuro, se espera probarla en estas condiciones de mejoramiento en algún proceso o etapa del desarrollo de un proyecto software. Es por ello que para este trabajo no se incluyeron ejemplos prácticos de métricas recolectadas. Finalmente, para este estudio de caso, el equipo de trabajo fue muy reducido, debido a las características del proyecto, es por eso que esta propuesta.

REFERENCIAS

- [1] M. Redondo. "Manual de calidad y procedimientos para la gestión del sistema de calidad de una empresa de desarrollo de software", pp. 15-25. Fecha de consulta: 24 de Mayo de 2016. URL: <http://bibing.us.es/proyectos/abreproy/30060/fichero/PROYECTO.pdf>
- [2] E. Diez. "Aseguramiento de la calidad en la construcción de sistemas basados en el Conocimiento: un enfoque práctico", Revista Latinoamericana de Ingeniería de Software, 1(5): pp. 167-206, ISSN 2314-2642, 2013.
- [3] G. Gómez. "Avances en las Mejoras de Procesos Software en las MiPyMes Desarrolladoras de Software: Una revisión Sistemática", Revista Latino Americana de Ingeniería de Software 2 (4): 262-268, ISSN 2314-2642, 2014.
- [4] J. Moreno. "Un Acercamiento a las Prácticas de calidad de software en las MiPyMESPS" Revista Lasallista de Investigación Vol. 7 N° I, pp. 17-24, ISSN: 1794-4449, 2010.
- [5] T. Grechenig and W. Zuser, "Creating organic software maturity attitudes (COSMA) selected principles and activities for software maturity in small and medium software enterprises," in Fourth International Conference on Quality Software, 2004, pp. 134-143.
- [6] I. Richardson, "Software process matrix: a small company SPI model. Software Process: Improvement and Practice," vol. 6, pp. 157-165, 2001.
- [7] M. E. Fayad, M. Laitinen, and R. P. Ward, "Software Engineering in the Small," Communications of the ACM, pp. 123-132, 2000.
- [8] Pressman. Ingeniería del Software. Un enfoque práctico", McGraw-Hill 7ta edición, México, 2010.

- [9] K. Ishikawa. "Que es el control de calidad" la modalidad japonesa (2a ed), Colombia: Norma. p. 13, 1986.
- [10] J. Juran y F. Gryna. "Análisis y planeación de la calidad", (3ers ed), Mexico: McGraw p. 5, 1998.
- [11] L. Cuatrecasas. "Gestión integral de la calidad. Implementación, control y certificación", ediciones gestion 2000, 3era edición, Barcelona, 2005.
- [12] ISO 9000:2005. "Sistema de gestión de la calidad Fundamentos y vocabulario".
- [13] A. Riveras. "Diseño de un sistema de gestión de Calidad para empresas de desarrollo de software en Venezuela, bajo la norma ISO 9001:2000", pp. 8-17, Fecha de consulta: 20 de Mayo de 2016. URL: http://tesis.ula.ve/pregrado/tde_busca/archivo.php?codArchivo=1269
- [14] A. Arias. "Gestión de la Calidad: Conceptos Básicos", pp. 4-7, Fecha de consulta: 15 de Mayo de 2016. URL: <http://pendiente demigracion.ucm.es/centros/cont/descargas/documento10123.pdf>
- [15] F. López. GitHub como herramienta docente, XXI Jornada de la Enseñanza Universitaria de la Informática, Andorra la Vella, Julio del 2015, ISBN: 978-99920-70-10-9.
- [16] P. Pesado. Calidad en el desarrollo de sistemas de software, WICC 2010 - XII Workshop de Investigación de Ciencias de la Computación, 2010.
- [17] G. Aguirre. "Desarrollo de un marco metodológico del proceso de verificación y validación de software para pequeñas y medianas empresas". Revista de la Facultad de Ingeniería Industrial, 18(2): pp. 145-154, 2015.
- [18] G. Paladines, "Administración de calidad en el desarrollo de un sistema de información". Facultad de ingeniería en electricidad y computación (FIEC). 2014.
- [19] Ishikawa K., & Union of Japanese Scientists and Engineers, QC Circle Headquarters, "How to Operate QC Circle Activities", Tokyo: Union of Japanese Scientists and Engineers (JUSE). 2003.
- [20] N. Meléndez. Propuesta de una guía para la incorporación de enfoques de gestión de calidad total a ingeniería de requisitos, universidad católica del norte, Chile, 2011.
- [21] N. Meléndez. "IR-SIXSIGMA: Mejora de Calidad en Ingeniería de Requisitos Mediante la Aplicación de Metodología Six-Sigma". Workshop Internacional EIG, diciembre del 2009.
- [22] N. Meléndez. "Aplicación de Sistemas de Codificación IR-Kanban para Mejoras en Trazabilidad de Requisitos Elicitados", IWASE, Temuco, Chile, 11-15 de Noviembre del 2013.
- [23] B. O'Sullivan. "Mercurial: The Definitive Guide". O'Really Media Inc, 2009.
- [24] E. Tello. Revisión de los sistemas de control de versiones utilizados en el desarrollo de software, Ing. USBMed, Vol. 3 N° 1. Enero-Junio 2012.

Copyright of INGENIARE - Revista Chilena de Ingeniería is the property of Universidad de Tarapaca and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.