 doi: https://doi.org/10.21500/22563202.4299

ISO 29110 en Colombia: de la teoría a la práctica

Manuel Pastrana[1]
Institución Universitaria Antonio José Camacho (Colombia)

Hugo Ordóñez[2]
Carlos Cobos[3]
Universidad del Cauca (Colombia)

Recibido: junio 15 de 2019. Revisado: agosto: 23 de 2019. Aceptado: noviembre 20 de 2019

Referencia norma APA: Pastran, M., Ordóñez, H., & Cobos, C. (2019). ISO 29110 en Colombia: de la teoría a la práctica. *Rev. Guillermo de Ockham, 17*(2), 71-80. doi: https://doi.org/10.21500/22563202.4299

Resumen

Las empresas desarrolladoras de software muy pequeñas o *Very Small Entities* (VSE), con menos de 25 empleados, se ven obligadas a garantizar la calidad del proceso para disminuir el reproceso e incrementar sus ganancias. Esto implica cumplir con buenas prácticas recomendadas en diversos estándares internacionales, que ofrecen pautas para mejorar los procesos y productos de software. Sin embargo, los requisitos de estos estándares son difíciles de cumplir para las VSE por su capacidad y los costos de implementación. Este documento describe el proceso de implementación de la norma ISO 29110 en el contexto específico de la región del Valle del Cauca en Colombia, para cuatro empresas, y expone las buenas prácticas, las herramientas y las técnicas utilizadas con resultados prometedores.

Palabras clave: ISO 29110, desarrollo de software, buenas prácticas.

ISO 29110 in Colombia: from theory to practice

Abstract

Today, the Very Small Entities (VSE), companies with less than 25 employees, are forced to guarantee the quality of the process to reduce reprocessing and increase their profits. This implies the use of good practices recommended in various international standards which offer guidelines to improve software processes and products. However, the requirements of these standards are difficult for VSEs to meet because of their capacity and implementation costs. This document describes the process of implementations of ISO 29110 in the specific context of the Valle del Cauca region in Colombia for 4 companies, exposing the good practices, tools and techniques used.

Keywords: ISO 29110, software development, good practices.

1. Institución Universitaria Antonio José Camacho, Cali, Colombia. mapastrana@admon.uniajc.edu.co
2. Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Popayán, Colombia. hugoordones@unicauca.edu.co
3. Universidad del Cauca, Popayán, Colombia. ccobos@unicauca.edu.co

Introducción

Las multinacionales proveedoras de software ofrecen sus soluciones y compiten constantemente por destacar en el mercado a través de un factor diferenciador clave, a saber, la calidad. Estas empresas requieren operaciones descentralizadas en el exterior con el fin de reducir costos y acceder a mano de obra calificada, sin tener que cubrir gastos de viáticos extensos por proyecto (Beecham, O'Leary, Richardson, Baker, & Noll, 2013), por lo que en su mayoría deciden tercerizar servicios con empresas de menor tamaño, siempre y cuando los costos de transacción no perjudiquen la calidad, la seguridad del proceso y la confiabilidad del producto final (Annous, Livadas, & Miles, 2010).

En este sentido, para acceder al mercado de la tercerización de servicios software, las empresas con 25 empleados o menos (VSE), deben garantizar un alto grado de madurez y calidad en sus procesos. Para esto, en algunas ocasiones hacen uso de estándares de procesos de software tales como ISO/IEC 12207, ISO/IEC 15504 o modelos robustos como CMMI, que proporcionan un conjunto de buenas prácticas y pautas para mejorar la calidad del proceso de software y productos resultantes de ese proceso. Adicionalmente, las personas que laboren en estas empresas deben cumplir con una formación que les permita adaptarse fácilmente a estos modelos. Esto no necesariamente sucede de manera natural en el contexto colombiano, de acuerdo con lo mencionado en Grass Ramírez, Collazos Ordóñez, & González González (2017).

Sin embargo, la percepción de las pequeñas empresas para la adopción e implementación de estos estándares es negativa. Por un lado, estos generan mucha documentación y burocracia, lo cual no es favorable por el aumento de los costos operativos de las VSE. Por otro lado, algunas prácticas no son aplicables a proyectos pequeños que son los más comunes para este tipo de compañías. De alguna manera, estos modelos dejan entrever demasiada exigencia y autoridad y alto costo de evaluación, como es el caso de CMMI, que incluyen esfuerzo, tiempo y dinero no fácilmente asumible por este tipo de empresas. El factor cultural también interviene, dado que se requiere una forma distinta de pensar y que el personal haga otro tipo de tareas diferentes a las que está acostumbrados, lo que también genera un cierto rechazo a su adopción. Además, algunos de sus integrantes comparten varias funciones, entre ellos el administrador, quien en muchos casos es responsable de más de un proyecto o puede tener responsabilidades tanto técnicas como administrativas (Ehsan, Perwaiz, Arif, Mirza, & Ishaque, 2010).

Sobre la base de que a nivel mundial los productos de software de las VSE son muy importantes para la economía, la ISO y el comité técnico conjunto para estándares de T.I (JTC1) definieron la norma ISO/IEC 29110 (de aquí en adelante ISO 29110, como normalmente se le conoce) con el fin de brindar mecanismos de estandarización de buenas prácticas que les permita su inclusión y competitividad en el mercado global (Laporte & Connor, 2016). ISO 29110 define el conjunto de prácticas que se deben seguir y los documentos mínimos necesarios para contar con un proceso de desarrollo basado en la calidad de sus entregables. Adicionalmente, introduce prácticas, herramientas y técnicas que apoyan el ciclo de desarrollo y los conceptos de estandarización requeridos para una alta calidad (Muñoz, Mejía, Lagunas, Investigación, & Zacatecas, 2018).

En este artículo se propone un modelo para la implementación de la norma ISO 29110 a partir de las prácticas efectuadas con cuatro empresas del Valle del Cauca, Colombia. El modelo está fundamentado en las buenas prácticas que se deben llevar a cabo en cada uno de los elementos que componen la norma 29110, que son: ingeniería de requisitos, gestión de la configuración, arquitectura funcional y física, verificación y validación, integración, despliegue del producto, gestión de interfaz, auto evaluación y gestión de proyectos.

El resto del documento se encuentra organizado de la siguiente manera. En primer lugar se tiene el escenario de motivación, seguidamente la descripción del modelo propuesto y sus pasos, luego los resultados de estudio de la aplicación del modelo en un conjunto de empresas seleccionadas y finalmente se presentan las conclusiones y el trabajo por desarrollar en el futuro.

Escenario de motivación

El sector TI cumple un papael importante en el desarrollo social y económico de todo país al generar constantemente soluciones a diversos problemas. Por este motivo, este servicio es ofertado por empresas de todo tamaño, lo que impone un mercado tan amplio que es posible que todos compitan en él sin que haya un monopolio visible de una sola empresa que domine por encima de las otras. Factores como la calidad, el precio, la velocidad de trabajo y la capacidad de trabajo, aumentan la credibilidad de cada empresa y son relevantes a la hora de tomar decisiones para contratar. Este panorama se amplía mucho más cuando la globalización entra a formar parte de las variables que

mueven el mercado, permitiendo con ello que oferentes y compradores de todo el planeta interactúen.

En Colombia, específicamente en el departamento del Valle del Cauca, se encuentra que el 95 % del total de empresas están catalogadas como VSE, según el informe de caracterización de la industria del software de SeniSoft y Fedesoft (2017). Adicionalmente, el mismo informe afirma que la mayoría de los compradores son internacionales y provienen de Estados Unidos, México, Costa Rica y Ecuador. Es importante resaltar que el 80 % de los compradores que han adquirido un servicio en Colombia dentro de este marco, buscaban productos de software para el sector financiero y gestión de recursos o desarrollos a la medida (*tailor-made software*). El restante 20 % requería contenidos digitales como animaciones, videojuegos y comerciales de televisión.

Para que las VSE lleguen y se sostengan en estos mercados internacionales, deben demostrar un proceso de desarrollo basado en estándares reconocidos por la industria que permitan obtener productos y ofrecer servicios software de alta calidad.

Norma ISO 29110

Para lograr la implementación de la norma ISO 29110, se propone abordar el modelo de la siguiente manera. Primero, dividir el trabajo por las categorías que define la norma. Segundo, categorizar los perfiles genéricos con los cuales clasificar las empresas. Tercero, clasificar las empresas de acuerdo con los perfiles propuestos por cada categoría. Cuarto, capacitar a las empresas en asuntos que presenten mayor deficiencia. Y quinto, exponer dentro de cada categoría un grupo de estrategias para alcanzar el siguiente nivel de la clasificación o superior.

Categorías que define la norma

La ISO 29110 define nueve categorías para ser abordadas e fin de mejorar la operación de las VSE (IEEE, 2011). En la Figura 1 se puede apreciar estas categorías.

Figura 1
Categorías norma ISO 29110. Fuente ISO/IEC TR 29110-5-1-2. Guía de gestión e ingeniería

Análisis de requerimientos

El trabajo (Pressman & Maxim, 2015) describe esta etapa del proceso de desarrollo de software como el momento en el que se recopilan, interpretan y refinan las necesidades o requisitos de un proyecto, de manera que la visión de la solución por construir sea la misma para todos los interesados. Es una de las etapas más críticas del proceso y de las que menos atención recibe en las VSE según (Hastie & Wojewoda, 2015). Algunas empresas no le dan importancia al formalismo de los requisitos a través de artefactos como los casos de uso o las historias de usuario, bajo modelos como los propuestos por Beck,(1999) y Cohn (2004).

Gestión de proyectos

El objetivo principal de esta área es hacer una proyección del presupuesto en tiempo y dinero necesario para cumplir y construir la solución solicitada (*Project Management Institute*, 2013), gestionando todos los artefactos y mecanismos necesarios para su cumplimiento y haciendo el debido seguimiento para tomar medidas correctivas en caso de presentarse un inconveniente con lo planeado (PMI, 2004). Para cumplir con lo mencionado, es necesario generar un canal de comunicación continuo que informe los resultados del seguimiento y el control del proyecto, gestione los riesgos adecuadamente y haga cumplir el cronograma de compromisos en su totalidad (*Project Management Institute*, 2013).

Arquitectura y diseño detallado

Esta categoría podría verse reflejada en la etapa de diseño dentro del ciclo de desarrollo descrito por Pressman & Maxim (2015), en el que el resultado final no debe ser un producto empírico basado únicamente en la experiencia. Las soluciones a nivel del diseño arquitectural implican un conocimiento claro sobre qué atributos de calidad (patrones de diseño) deben ser satisfechos para lograr el comportamiento adecuado del sistema frente a ellos. Estos patrones pueden variar considerablemente según si las soluciones son desarrollos tipo escritorio, web, web responsive, móvil o de integración. De acuerdo con Hastie

& Wojewoda (2015), es una de las etapas más descuidadas por falta de conocimiento y por su alta complejidad.

Verificación y validación

Beck (1999) expresa que toda empresa de desarrollo de software debe tener, como buena práctica, un estándar de código que la guíe para una programación correcta a nivel sintáctico. Adicionalmente, debe ser consciente de las malas prácticas recurrentes dentro de su forma de codificación, como son el uso excesivo de condiciones anidadas, variables definidas sin uso, definición de constantes cuando son necesarias, etc. Estas malas prácticas llevan a que el equipo de trabajo deba hacer inspecciones periódicas del código, sean manuales o automatizadas por medio de analizadores estáticos de código. Lo anterior, busca identificar previamente a las pruebas funcionales todos los posibles inconvenientes que se presentan dentro del desarrollo y que no dependan de la lógica funcional del sistema.

Construcción y pruebas unitarias

La etapa en la que los equipos de desarrollo pasan la mayor parte del proyecto es la de construcción (Pressman & Maxim, 2015). Los requisitos de software obtenidos en la etapa de análisis y que toman forma en una solución mediante los planteamientos de diseño de software de la etapa de diseño, son construidos aquí. Por lo anterior, las buenas prácticas dentro de la cultura de desarrollo de software deben generar un sello de calidad propio del equipo, que permita cumplir con las expectativas de todos los interesados antes de poder indicar que cada meta dentro del ciclo de desarrollo, es cumplida (Schwaber & Sutherland, 2014). La mejor forma de lograr esto según Beck (1999) es a través de las pruebas unitarias. Estas permiten que los desarrolladores, una vez construyen una unidad de código, puedan revisar si funcionalmente cumple con lo requerido (correctitud). Además, pueden verificar que el software responda correctamente frente a errores sin que el sistema pierda operatividad.

Integración y pruebas

La integración continua es una buena práctica, necesaria para que el equipo de desarrollo esté preparado para desplegar en un ambiente final o, en su defecto, en una réplica de este (Beck, 1999), desde el primer momento en que aborda la etapa de construcción. De este modo, el equipo no se verá frente a situaciones posteriores en las que el producto no pueda ser desplegado en el servidor por problemas de configuración. Esta práctica va muy de la mano con el despliegue continuo también propuesto por Beck (1999).

Despliegue del producto

El despliegue del producto debe ser una parte natural y constante dentro del proceso de desarrollo de software como propone Beck (1999) en el modelo XP. A medida que el equipo de trabajo está familiarizado con el ambiente final, más certeras serán las pruebas ejecutadas que garanticen la calidad del entregable, minimizando así los problemas de la salida a producción.

Autoevaluación

Las reflexiones retrospectivas inherentes a los modelos agiles como XP y SCRUM, hacen una gran diferencia a la hora de implementar la mejora continua como parte de la cultura organizacional. A medida que las empresas sean conscientes de qué han hecho bien para seguirlo haciendo y qué han estado ejecutando mal para cambiarlo, el reproceso por malas prácticas se verá disminuido. Si bien (PMI, 2004) tiene un formato denominado lecciones aprendidas, la mayoría de las VSE que llevan este marco a su forma de trabajo solamente llenan el documento al final del proyecto, sin hacer un plan de mejoras para nuevos proyectos, quedando esta información en el olvido y sin generar impacto alguno.

Versionamiento

Por otra parte, Beck (1999) también propone el versionamiento de código como una buena práctica de desarrollo de software. Aquí el equipo unifica su trabajo a través de un repositorio de código centralizado, en el que todos los cambios del producto en construcción o terminado son recolectados. Este repositorio permite llevar una auditoría de cambios, prevenir pérdida de información y recuperación ante fallos por descuidos del equipo sobre el código fuente en constante evolución. Es importante resaltar que Beck (1999) ve las herramientas de versionamiento como un apoyo a la buena práctica de propiedad colectiva del código, donde la fuente está disponible en cualquier momento para todo el equipo.

Categorizar los perfiles genéricos con los cuales clasificar las empresas

La norma ISO 29110 expone cuatro perfiles con los que se puede clasificar el estado de implementación de buenas prácticas en cualquier empresa, estos son:

– **Inicial**. Indica que la empresa trabaja de manera completamente empírica. No hay estándar de desarrollo ni gestión de proyectos y las etapas del ciclo de desarrollo se hacen sin seguir ningún lineamiento.

– **Básico**. Las empresas tienen un formalismo para sus distintas etapas, pero no implementan estándares internacionales aceptados por la industria, ni buenas prácticas.

– **Intermedio**. Las empresas conocen y usan algunos estándares y buenas prácticas.

– **Avanzado**. Las empresas conocen muy bien el proceso de desarrollo de software interno gracias a su estandarización. Todos sus artefactos están basados en modelos aceptados y reconocidos internacionalmente y el uso de buenas prácticas es parte de la cultura organizacional.

De la teoría a la práctica

La norma ISO29110 va muy de la mano con la filosofía de los *frameworks* ágiles como SCRUM y XP. Por un lado, pretende que la documentación sea ligera, sencilla de diligenciar y muy útil para el equipo de trabajo. Por otro, busca que toda la información generada permita hacer el trabajo con el máximo sello de calidad posible y minimizar el reproceso mediante los controles de calidad requeridos por cada fase.

Para el caso de estudio se hizo una convocatoria a las empresas del sector en Cali, en total 19. De este total se seleccionaron cuatro que serán denominadas a partir de aquí EMP1, EMP2, EMP3 y EMP4. La selección de estas empresas se dio con el cumplimiento de los siguientes requisitos (teniendo en cuenta que debían entregarse en un plazo de tiempo máximo):

– Certificado de Cámara de Comercio con vigencia máxima de tres meses.

– Hojas de vida del equipo propuesto para el desarrollo del proyecto, con los respectivos soportes de formación y experiencia. Para esto se definió un formato específico.

– Formato de autorización de uso y almacenamiento de datos, uno por cada persona involucrada en el proyecto.

– Carta de autorización de uso y almacenamiento de datos por parte de cada empresa.

Para iniciar la implementación del modelo se generó un instrumento de evaluación tipo encuesta sobre la implementación de buenas prácticas a nivel de las compañías. Esto permitió la categorización y la clasificación de perfiles genéricos de acuerdo con lo que sugiere la norma ISO 29110. Este instrumento buscó evaluar el uso de cada uno de los estándares de la norma. Lo anterior permitió, además, identificar las áreas que requerían capacitación, sobre qué asuntos y en qué medida. El resultado de la evaluación se puede observar en Tabla 1.

Tabla 1
Clasificación de categorías por perfiles según norma ISO 29110

Categorías	EMP1	EMP2	EMP3	EMP4
Análisis de requerimientos (1)	Básico	Básico	Básico	Básico
Gestión de proyectos (2)	Básico	Básico	Intermedio	Inicial
Arquitectura y diseño detallado (3)	Básico	Básico	Intermedio	Básico
Verificación y validación (4)	Básico	Básico	Intermedio	Intermedio
Construcción y pruebas unitarias (5)	Básico	Inicial	Intermedio	Básico
Integración y pruebas (6)	Básico	Inicial	Intermedio	Inicial
Despliegue del producto (7)	Básico	Inicial	Básico	Básico
Autoevaluación (8)	Inicial	Inicial	Inicial	Inicial
Versionamiento (9)	Inicial	Básico	Intermedio	Inicial

Esta caracterización permitió identificar las áreas más débiles y definir las estrategias específicas que se van a aplicar por cada categoría, buscando con ello que las empresas alcancen la siguiente clasificación. A continuación, se presentan las prácticas efectuadas por cada área de la norma iso 29110 en las cuatro empresas seleccionadas.

Análisis de requerimientos

El trabajo empírico es el principal generador de reproceso porque dificulta crear un lenguaje común entre los interesados en el proyecto y distorsiona la visión del producto. Aquí se estandarizó con cada compañía el artefacto con el que se elicita la información. De acuerdo con Hastie & Wojewoda (2015), los modelos ágiles tienen un mayor impacto en el éxito de los proyectos. Por lo anterior, el uso de historias de usuario se recomendó a todas las empresas. Si bien no existe un estándar para este artefacto, sí hay modelos como el propuesto por Cohn (2004) que hacen simple la labor de crear un lenguaje común entre usuarios y equipo de desarrollo. El modelo propuesto por Cohn sugiere el uso de la siguiente frase: yo como [papel] + requiero [funcionalidad] + para [motivo] + con [estos criterios de aceptación]. Esta frase simplifica la escritura

de la historia de usuario permitiendo saber quién ejecuta la acción (papel), qué funcionalidad se espera ejecutar, qué valor le entrega al proyecto (motivo) y cómo compruebo que la historia fue desarrollada completamente (criterios de aceptación). En la primera se explica que las primeras tres partes de la frase yo como [papae] + requiero [funcionalidad] + para [motivo] responden a la denominación de historia de usuario épica según Cohn (2004). Aquí se aprendió a dominar este concepto y a elaborarlas correctamente, manteniendo la claridad sobre el motivo y la atomicidad de las historias (que respondan a una sola funcionalidad). La segunda sesión buscó apropiar el concepto y la forma de escribir los criterios de aceptación. Las recomendaciones dadas para la escritura de estos consisten en ubicar dónde sucede la acción (pantalla), especificar los campos que se requieren utilizar en esa funcionalidad, si es posible de una vez indicar el tipo de dato, tamaño, restricciones no funcionales, mensajes de error o de éxito y llamados a otras historias en caso de ser necesario para mantener la atomicidad. Una vez puestos en marcha los conceptos aprendidos, las empresas desde una etapa inicial (primera entrevista con el usuario) pueden generar una descripción utilizando el mapa de historias (técnica que permite una división funcional por posibles módulos de un aplicativo) haciendo uso de la definición épica. Posterior a la detección del mínimo producto viable, que en otras palabras es lo que se alcanza a hacer en el tiempo y presupuesto esperado, el equipo puede entrar a construir el detalle de las historias de usuario complementándolas con los criterios de aceptación. Adicional a lo explicado en las dos sesiones, se diseñó un formato que facilita la aplicación de lo visto. En la Tabla 2 se aprecia un ejemplo del formato creado.

Tabla 2
Formato de *product backlog* compuesto por historias de usuario. Criterios de aceptación (CA)

ID	Papel	Funcionalidad	Motivo	CA	Observaciones
HU_N	Yo como [papel]	Requiero [funcionalidad]	Para [motivo]	Con [estos criterios de aceptación]	Dudas o aclaraciones.

Gestión de proyectos

Para hacer una correcta gestión de proyectos es necesario partir de un buen análisis de requisitos, consolidado en este modelo a través de un product *backlog*. Hay varias buenas prácticas que se pueden tomar de lo propuesto en Schwaber & Sutherland (2014). Primero, para delimitar el alcance real del proyecto se debe definir claramente la prioridad de atención de las historias de usuario y luego estimar la complejidad asociada a estas para fijar una planeación de fechas de entrega. Por lo anterior, el formato de product *backlog* propuesto en la sección anterior sufre una modificación que permite adicionarle dos columnas que incluyen los valores correspondientes a la priorización y a la estimación de cada historia de usuario, que serán los objetivos de las dos sesiones realizadas con cada empresa (Tabla 3).

Para la primea sesión con las empresas se trabajó la priorización de las historias mediante la técnica MoSCoW, en la que se sugiere que el orden en la prioridad sea entregado por los interesados del proyecto y manejado de acuerdo con el valor de negocio que aporta. El detalle de las siglas MoSCoW se describe a continuación:

M = Must Have. Los interesados manifiestan que esta funcionalidad debe estar. Es parte del core del aplicativo y aporta mucho valor.

S = Should Have. Los interesados manifiestan que esta funcionalidad si bien no es parte del core, aporta valor a los procesos secundarios que utiliza el aplicativo.

C = Could Have. Los interesados manifiestan que, si bien no es importante, sería bueno tenerlo en algún momento.

W = Won't have by now. Los interesados manifiestan que no es necesario tener esta funcionalidad, al menos por ahora.

Cada historia de un proyecto es priorizada de este modo, así el equipo puede saber el orden de atención de las necesidades y dividirlas en periodos de tiempo que van de una a cuatro semanas denominado *sprint*. Adicionalmente, así como el total de historias de usuario de un proyecto se consolida en un product *backlog* (Schwaber & Sutherland, 2014). Se sugiere que el trabajo que se va a hacer por sprint se separe en un artefacto denominado *sprint backlog* para garantizar un fácil seguimiento de las metas propuestas por *sprint*.

Tabla 3
Formato *Product Backlog* para las empresas. Criterios de aceptación (CA), priorización (prio), estimación (estim), observación (obs)

ID	Papel	Funcionalidad	Motivo	CA	Prio	Estim	Obs
HU_N	Yo como [papel]	Requiero [funcionalidad]	Para [motivo]	Con [estos criterios de aceptación]	MSCW	Complejidad o peso de la historia	Dudas o aclaraciones.

Durante la segunda sesión, según Schwaber & Sutherland (2014), se sugiere que el equipo y todos los interesados deben estar informados constantemente de la evolución de las actividades de desarrollo involucradas dentro del *sprint*. Aquí se explica cómo crear y manejar un tablero *scrum*. Una forma fácil de implementar estos tableros es utilizando la herramienta Trello. En esta herramienta se pueden crear tres columnas con la definición base propuesta en los modelos agiles: pendientes (*to do*), en ejecución (*doing*), terminado (*done*). La Figura 2 muestra un ejemplo base de un tablero en Trello bajo la recomendación de Schwaber & Sutherland (2014):

Figura 2
Tablero Scrum para informe de avances

Arquitectura y diseño detallado

Las empresas contaron con tres sesiones utilizando como base el trabajo de Kruchten (2006) para la construcción del documento de arquitectura de software (DAS) como artefacto principal de diseño.

El DAS se divide en dos partes. La primera consiste en la identificación de los atributos de calidad y patrones de diseño que permiten guiar la arquitectura. La segunda es su representación a través de diagramas UML. La primera sesión con las empresas buscó apropiar el concepto de atributos de calidad que responden al comportamiento deseado frente a requerimientos del sistema, como son la portabilidad u la mantenibilidad, entre otros. Con la identificación correcta de los atributos en un proyecto de ejemplo, la sesión dos ayudó a las empresas a entender cómo hacer un mapeo base que permita identificar los patrones de diseño asociados a los atributos de calidad. Un patrón de diseño se puede considerar como una solución que ya existe y está comprobada para resolver un problema ecurrente. Algunos patrones son el singleton y la inyección de dependencias. Una vez apropiados los conceptos anteriores, en la tercera sesión se explica cómo diagramar las vistas sugeridas por Kruchten (2006) utilizando UML. La Tabla 4 muestra un ejemplo de mapeo básico elaborado en las empresas para proyectos en *Java* y *php* que sirve de base y les permite a las empresas que esta plantilla siga creciendo, evolucionando o variando de acuerdo con los retos a afrontar en cada proyecto.

Tabla 4
Formato atributos de calidad y patrones de diseño de una solución

Atributo de calidad	Descripción	Patrón de diseño	Donde se aplica
Confidencialidad	Es la ausencia de acceso no autorizado a la información	Autorizador-autenticador	Se implementa un *login*

Verificación y validación

Para garantizar el cumplimiento de los *sprints* es clave utilizar sistemas de retroalimentación basados en verificación y validación del avance del desarrollo (Pastrana, Ordoñez, Rojas, & Ordóñez, 2019). Lo primero que todas las empresas tuvieron que hacer fue adoptar un estándar de desarrollo de software, entenderlo y apropiarlo. En todas se implementó el estándar internacional de los lenguajes utilizados (dos empresas utilizan java, una php y una .net y *Java*). Con base en esto se puede lograr la propiedad colectiva del código propuesta por Beck (1999) y se logra minimizar la curva de aprendizaje asociada al soporte o al personal que ingresa nuevo a la empresa. Adicionalmente, para corroborar que la calidad del desarrollo a nivel funcional está cumpliendo con lo especificado en las historias de usuario de acuerdo con Pastrana *et al.* (2019) y pueda lograr correctamente el estado de terminado o *done* en el tablero informativo de avance, la implementación de pruebas unitarias se vuelve el sello de calidad y el control mínimo que debe tener cada desarrollador para identificar si lo construido cumple con los criterios de aceptación (esto se trató en otra sesión taller). Aunque pareciera que toma más tiempo el desarrollo al incluir esta práctica, los resultados indican que el reproceso tiende a disminuir y los requisitos desarrollados devueltos en revisiones posteriores, como las pruebas de aceptación, son mucho menores que si la práctica no se implementa.

Construcción y pruebas unitarias

Para una primera sesión se hizo una presentación de las buenas y malas prácticas de desarrollo de software de acuerdo con el estándar de programación para los lenguajes de programación *Java*, php y .net. Una vez presentados los conceptos previos se hizo un sondeo para identificar cuáles de estos son los más frecuentes dentro de cada empresa. Los casos más comunes fueron: no definir cadenas de texto de mensajes y etiquetas de campos en pantalla como constantes, sino escribirlos directamente en cada parte donde es requerido. Esto es ineficiente dado que

separan espacio de memoria cada vez que vuelven y crean un mismo texto. Por ejemplo, si se tienen 20 botones que dicen "guardar" no definen una constante para ser llamada en el valor de nombre del botón, sino que escriben en cada botón la misma cadena. Adicionalmente, el segundo caso más repetido fue el mal manejo de la complejidad ciclomática asociada a los ciclos anidados. En algunos casos, por la urgencia de salir rápido con el entregable no se hace una adecuada implementación de ciclos, lo que aumenta la complejidad de estos innecesariamente. Otros casos comunes fueron no definir nombres de variables claros, dejar librerías importadas sin ser usadas, declarar variables sin ningún uso y dejar métodos definidos sin usar. Lo más importante en esta etapa consistió en identificar lo que permite garantizar el cumplimiento de los *sprints* no solo en el aspecto funcional, sino en la alta calidad asociada con el desarrollo. Para lograr esto, es clave utilizar métricas de aceptación asociados a la calidad de lo que se está desarrollando. Esto se relaciona con las inspecciones automatizadas que pueden generarse con herramientas de análisis estático de código, pero también puede empezar a ejecutarse de manera manual mediante revisiones por pares. En la siguiente sesión se hizo un ejercicio de revisión por pares para identificar la importancia del estándar y revisar, mantener y modificar el código fuente ejecutado por otros sin tener mayores dificultades en la lectura y comprensión de este. Con esto claro, se procedió a entregar fuentes de diversos proyectos para revisarlos de acuerdo con el estándar definido e identificar si cumple o no, e identificar los casos más repetitivos para cada empresa.

Adicionalmente, en un último taller se revisó el asunto de prueba unitarias, considerados como la mínima prueba asociada a una unidad de código que permite identificar la correctitud y cómo este código reacciona a fallos. Aunque pareciera que toma más tiempo el desarrollo al incluir esta práctica, los resultados indican que el reproceso tiende a disminuir y los requisitos desarrollados devueltos en revisiones posterioresusando, por ejemplo, las pruebas de aceptación, son mucho menores que si la práctica no se implementa (Pastrana *et al.*, 2019).

Integración y pruebas

La integración continua del trabajo llevado a cabo por todos los colaboradores de un proyecto, permite identificar de manera inmediata cuándo un cambio en el código fuente versionado presenta un error sintáctico y puede afectar el desplegable de manera negativa (Pastrana *et al.*, 2019). Se utiliza la herramienta Jenkins para lograr de forma automática tomar los cambios de un versionador y ejecutar así las validaciones requeridas que identifiquen si el desplegable se puede generar o no. Por tanto, en la primera sesión se hizo un taller para entender cómo funciona la herramienta Jenkins, cómo crear un proyecto de integración continua en esta herramienta, cómo unirlo con el versionador de cada empresa y cómo configurar reglas de ejecución automática para facilitar la información. Por ejemplo, que cada hora revise la última versión que esta versionada o que notifique por correo cuando no se pudo generar el desplegable. En esta práctica se logró que las empresas entendieran la importancia de contar con esta herramienta que de manera automática identifica los cambios del código y cómo estos cambios pueden generar un impacto negativo en la evolución del proyecto si son detectados muy tarde. Por otra parte, esta misma herramienta tiene la posibilidad de poner en un servidor de pruebas el desplegable generado, de manera automática. A esto se lo conoce como despliegue continuo y también fue presentado en el taller. En una última sesión se organizó un taller sobre el uso de sonar para medir la calidad del desarrollo mediante pruebas estáticas de código que incrementan aún más la posibilidad de medir si el equipo de desarrollo está programando de acuerdo con el estándar de código y a las buenas practicas sugeridas.

Despliegue del producto

El despliegue del producto puede ser hecha de dos maneras. La primera es un proceso manual y la segunda automatizado. Para ambos casos se hizo un taller en el que el código fuente que se va a poner en producción debe haber pasado previamente por todos los controles de calidad sugeridos hasta este punto, de acuerdo con Pastrana *et al.*, 2019). Una vez el entregable fue puesto en el servidor de pruebas de cada empresa, se hizo un taller que permitió versionar en la sección *tag* el desplegable para una rápida recuperación en caso de fallos.

Autoevaluación

La mejora continua es fundamental para incrementar las capacidades productivas de una empresa y es alcanzable mediante la autoevaluación por técnicas retrospectivas. Schwaber & Sutherland (2014) sugieren que cada vez que se termine un *sprint* se debe planear una reunión que toma entre una y cuatro horas para socializar lo que ha hecho el equipo de manera positiva y ha ayudado a conseguir los objetivos, pero también se revisan los aspectos negativos que frenan o impiden conseguir objetivos o completar tareas dentro del *sprint* ejecutado. Para las cuatro empresas, se hizo una sesión en la que se utilizó la técnica estrella de mar. Esta técnica consta de cinco frases que cada persona

en la reunión debe responder. Estas son: hacer más de, seguir haciendo, hacer menos de, dejar de hacer y empezar a hacer. El objetivo de cada frase es identificar lo que el equipo hace para que aprenda de sus aciertos o fallos y sea capaz de reaccionar para el siguiente *sprint*. Para ejecutar la técnica, cada frase tiene un tiempo cronometrado por quien dirige la reunión. Si se tiene, por ejemplo, solo una hora, se divide la hora entre las cinco preguntas, lo que nos da un tiempo de doce minutos por cada pregunta. Adicionalmente, todas las ideas expuestas son socializadas al finalizar los doce minutos para retroalimentación de todos. Es importante resaltar que cada idea debe ir en un *postit*. Estos recortes de papel de aproximadamente 7, 6 cm x 7,6 cm limitan a quien escribe a expresar solo lo esencial y no generar detalles innecesarios. Todos los compromisos adquiridos durante la retrospectiva deben ser ejecutados en el siguiente *sprint* a fin de que tengan impacto real dentro del proyecto. La forma de validar esto es por medio del tablero *Scrum*. En cada empresa se hizo con dos proyectos terminado casi consecutivos para identificar qué tan común eran sus errores y generar una cultura de cambio basada en la autoevaluación.

Versionamiento

Siguiendo la recomendación de Pastrana *et al.* (2019) para el cumplimiento de los objetivos de un *sprint* a nivel de desarrollo, la gestión de la configuración es clave para organizar, sincronizar y medir el trabajo del día a día durante un proyecto. Por lo anterior, en la primera sesión de cada empresa se indicó que todo el trabajo ejecutado por diversos integrantes de un equipo debe estar sincronizado mediante un repositorio central de versionamiento, para contar con un histórico de los cambios hechos al código fuente y su recuperación ante la presencia de fallos, además de tener las versiones finales puestas en producción congeladas y listas para usar cuando sea requerido. Adicionalmente, se explicó durante la misma sesión en cada empresa, que la estandarización del código es la clave para que todos los miembros del equipo entiendan lo escrito por otros y permite mantener el orden del trabajo desarrollado. Finalmente, se trabajó un taller que permitió utilizar la herramienta de versionamiento *Bitbucket*. El taller buscó entender que la herramienta posee una división en rama máster, ramas secundarias, *tag*, *bug* y *hotfix*. La rama principal o máster es la rama donde solamente está sincronizada la última versión estable que lista para ser puesta en producción. Esta rama solo puede ser tocada por el gestor de la configuración, quien hace la validación con herramientas complementarias para identificar si se cumple con los estándares de calidad y se pueden sincronizar los cambios de la rama máster. Cada desarrollador trabaja sobre una rama propia llamada *develop*. En esta rama todos los del equipo de desarrollo sincronizan sus cambios locales con los de los demás. Esta rama es afectada por dos variantes. Cuando alguien del equipo de desarrollo detecta en tiempo de desarrollo un error, lo cataloga en *hotfix* y lo resuelve. Esto indica que antes de seguir creciendo los cambios de *develop*, esto se ajusta y luego se continúa con el trabajo normal. El segundo caso ocurre cuando un error se produce y es detectado ya estando sincronizada la rama *develop*. Esta situación se cataloga como *bug*, se corrige y nuevamente se sincroniza con la rama *develop* para estabilizarla. Una vez la rama *develop* es estable y está sincronizada con la rama máster, se crea una copia del desplegable en una rama llamada *tag*.

Al finalizar el proceso de capacitación y acompañamiento a las empresas, se aplicó el mismo instrumento de evaluación de las buenas prácticas en las compañías para categorizar y clasificar el nuevo perfil, de acuerdo con lo que sugiere la norma ISO 29110. El resultado de la evaluación final se puede observar en Tabla 5. Al comparar con los resultados de la encuesta inicial presentada en la Tabla 1, se aprecia que todas las empresas mejoraron en todas las categorías y que dicho cambio implicó en algunos casos, cambios de valoración significativos. Por ejemplo, de básico a avanzado, así como otros fueron un poco más progresivos como de básico a intermedio.

tabla 5

Clasificación de categorías por perfiles según norma ISO 29110

Categorías	EMP1	EMP2	EMP3	EMP4
Análisis de requerimientos (1)	Avanzado	Avanzado	Avanzado	Avanzado
Gestión de proyectos (2)	Avanzado	Avanzado	Avanzado	Intermedio
Arquitectura y diseño detallado (3)	Intermedio	Intermedio	Avanzado	Intermedio
Verificación y validación (4)	Intermedio	Intermedio	Avanzado	Avanzado
Construcción y pruebas unitarias (5)	Intermedio	Intermedio	Avanzado	Intermedio
Integración y pruebas (6)	Intermedio	Intermedio	Avanzado	Básico
Despliegue del producto (7)	Intermedio	Intermedio	Avanzado	Intermedio
Autoevaluación (8)	Intermedio	Intermedio	Intermedio	Básico
Versionamiento (9)	Avanzado	Avanzado	Avanzado	Básico

Conclusiones

La adopción de la norma ISO 29110 permitió a las cuatro empresas organizar mejor su forma de trabajo, mantener el orden de su información en todo momento, tener trazabilidad de los proyectos y apropiar herramientas de control de calidad que reducen el reproceso.

Adicionalmente, mejoró la forma como recopilan la información, garantizando con ello un mejor resultado en los compromisos que pueden ser asumidos no desde la suposición, sino desde datos reales basados en estimación y priorización. Por otra parte, la implementación de las retrospectivas hace que los equipos tengan la posibilidad de entender qué hacen bien y mal para generar así una trasformación real en las compañías.

Como trabajo futuro, se espera a corto plazo complementar el trabajo en cada una de las fases de la norma 29110 con un entorno DevOps, que permita hacer la tarea de desarrollo y despliegue de forma colaborativa, en la que cada uno de los integrantes del grupo de desarrollo trabajen en pro de garantizar el éxito en todo el proceso de construcción de las aplicaciones desarrollas por cada empresa.

Referencias

Annous, H., Livadas, L., & Miles, G. (2010). OffshoreQA: A framework for helping software development outsourcing companies comply with ISO 9001:2008. Proceedings - 5th International Conference on Global Software Engineering, ICGSE 2010. https://doi.org/10.1109/ICGSE.2010.43

Beck, K. (1999). Extreme Programming Explained: Embrace Change. In XP Series. https://doi.org/10.1136/adc.2005.076794

Beecham, S., O'Leary, P., Richardson, I., Baker, S., & Noll, J. (2013). Who are we doing Global Software Engineering research for? Proceedings - IEEE 8th International Conference on Global Software Engineering, ICGSE 2013. https://doi.org/10.1109/ICGSE.2013.14

Cohn, M. (2004). User Stories Applied: For Agile Software Development (Addison Wesley Signature Series). In Writing (Vol. 1). https://doi.org/10.1017/CBO9781107415324.004

Ehsan, N., Perwaiz, A., Arif, J., Mirza, E., & Ishaque, A. (2010). CMMI/SPICE based process improvement. 5th IEEE International Conference on Management of Innovation and Technology, ICMIT2010. https://doi.org/10.1109/ICMIT.2010.5492803

Grass Ramírez, B., Collazos Ordóñez, C., & González González, C. (2017). "Propuesta de incorporación de competencias de formación en ingeniería". En *Guillermo de Ockham: Revista Científic*a, 15(1), 13. https://doi.org/10.21500/22563202.3188

Hastie, S., & Wojewoda, S. (2015). Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. Retrieved from http://www.infoq.com/articles/standish-chaos-2015 website: http://www.infoq.com/articles/standish-chaos-2015

IEEE. (2011). International Standard ISO / IEC / IEEE Systems and software engineering — engineering. ISO/IEC/IEEE 12207, 2011, 1–94. https://doi.org/10.1080/21670811.2017.1279978

Kruchten, P. (2006). Planos arquitectónicos: el modelo de "4 + 1". Vistas de la la arquitectura del software. IEEE Software, 12(6), 1–16.

Laporte, C. Y., & Connor, R. V. O. (2016). Implementing Process Improvement in Very Small Enterprises with ISO / IEC 29110 A Multiple Case Study Analysis. https://doi.org/10.1109/QUATIC.2016.27

Muñoz, M., Mejía, J., Lagunas, A., Investigación, C. De, & Zacatecas, A. C. U. (2018). Implementación del estándar ISO / IEC 29110 en entornos ágiles. Una revisión sistemática de literatura Implementation of the ISO / IEC 29110 standard in agile environments : A systematic literature review. 13th Iberian Conference on Information Systems and Technologies (CISTI). Retrieved from https://ieeexplore.ieee.org/document/8399332

Pastrana, M., Ordoñez, H., Rojas, A., & Ordoñez, A. (2019). Ensuring Compliance with Sprint Requirements in SCRUM: Preventive Quality Assurance in SCRUM. Advances in Intelligent Systems and Computing, 924, 33–45. https://doi.org/10.1007/978-981-13-6861-5_3

PMI. (2004). "PMI Fundamentos para la dirección de proyectos (guía del pmbok)". In *Fundamentos*. https://doi.org/10.15611/ie.2014.1.14

Pressman, R. S., & Maxim, B. R. (2015). Software Engineering : A Practitioner's Approach, Eighth Edition. In ACM SIGSOFT Software Engineering Notes. https://doi.org/10.1145/1226816.1226822

Project Management Institute. (2013). "Guía de los fundamentos para la dirección de proyectos". In *Global Standard* (Vol. 87). https://doi.org/HD69.P75G845 2013 658.4'04--dc23 2012046112

Schwaber, K., & Sutherland, J. (2014). *La guía definitiva de Scrum. Las reglas de juego*. Scrum.Org.