Implementation of Software Quality Control in E-Learning development projects: TEC Digital

Krissia Gómez-Román
TEC Digital
Instituto Tecnológico de Costa Rica
Email: kgomez@itcr.ac.cr

Ederick Navas
TEC Digital
Instituto Tecnológico de Costa Rica
Email: enavas@itcr.ac.cr

***Abstract*—The quality control in software applications that support the teaching-learning process is of great impact to the end user and their academic success. This paper describes the proposal of a quality control model called TD-CCS, implemented in a Learning Management System dotLRN, adapted for the development of applications in TEC-Digital. For measurement purposes, a data analysis was applied to ten projects in TEC-Digital, using evaluation criteria, aspects of documentation, structural quality of packages, functionality, web navigability, security and performance, integrity and compatibility. From the study it was obtained that in the test stage a total of 454 findings were captured, reflecting the importance of optimizing software quality in the early stages, as well as the need to define objective metrics that allow an integral quantitative analysis.**

***Keywords*—Quality Control, LMS, Methodology**

1. Introducción

El proceso de desarrollo de software busca la automatización de soluciones mediante la creación de sistemas informáticos, que respondan a las necesidades propias de cada organización y sus usuarios. Por esto, las actividades de control y aseguramiento de calidad de software cobran importancia dentro de los procesos de la organización.
Según Pressman, la calidad del software es "la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" [1]. En torno a esta definición es comprensible la criticidad de las actividades de calidad en el desarrollo de software, para la disminución de riesgos y errores.
Estas actividades varían de acuerdo a la organización, ambiente de desarrollo y necesidades propias. Es así como, los sistemas de gestión de aprendizaje, LMS[1], se han incorporado en la búsqueda de herramientas y actividades que garanticen la calidad en sus productos, de acuerdo con su metodología de trabajo basada en la colaboración y comunicación, para el aprendizaje.

1. Learning Management System

El TEC Digital es una dependencia que busca la implementación del e-learning respondiendo a las necesidades del Tecnológico de Costa Rica (TEC). Esto le ha permitido posicionarse como una plataforma de herramientas de tecnologías de información en la Docencia [2].
Actualmente, el TEC Digital ha incrementado su portafolio de proyectos, lo que demanda un proceso de implementación depurado, para garantizar una mejora continua en la calidad, y poniendo en manifiesto ciertas deficiencias en cuanto a gestión del software, que repercute en aspectos de seguridad, rendimiento y errores de funcionalidad para el usuario final.
Este artículo describe la metodología del modelo utilizado para el control de calidad de software en proyectos de desarrollo de e-learning, específicamente, en el TEC Digital. A este modelo junto con sus respectivas herramientas de aplicación se le denomina TD-CCS (TEC Digital - Control de Calidad del Software).

1.1. Descripción de la necesidad

Inicialmente, en el TEC Digital las pruebas de calidad aplicadas al software desarrollado eran realizadas por cada programador, sin embargo, era frecuente la presencia de errores en el producto final, produciendo disconformidad por parte de los usuarios. Adicionalmente, bajo esta modalidad de trabajo no se contaba con registros para efectos de estadísticas de hallazgos y mejoras realizadas, ni trazabilidad en cuanto a los esfuerzos llevados a cabo.
Posteriormente, esta actividad pasa a ser ejecutada por personas diferentes de aquellas que elaboran el producto, quienes asumen el rol de *Gestores de Calidad*. Se busca que el encargado del proceso funcione como un representante del cliente en el interior de la empresa [3].
Posteriormente y con el crecimiento del portafolio de proyectos del TEC Digital, se vuelve esencial desarrollar o adoptar un recurso que sirva como herramienta para estandarizar los aspectos considerados para cada revisión, así como para llevar trazabilidad de hallazgos y seguimiento sobre los mismos, asegurando que sean gestionados antes de su liberación al ambiente de producción.
A raíz de estas necesidades surge la propuesta del modelo de control de calidad dentro del TEC Digital, y conocido como modelo TD-CCS, del cual se profundiza en las

siguientes secciones.

2. Trabajo relacionado

Existen diferentes modelos sobre la calidad de software, y a partir de éstos se definen las primeras características de calidad. Algunas de estas características son: confiabilidad, mantenibilidad, funcionalidad, entre otras, que varían entre los diversos modelos que se analicen.
De esta forma [4] hace un abordaje de algunos modelos, tales como modelo de McCall, FURPS (Funcionality Usability Reliability Performance y Supportability, de aquí su nombre), Dromey, SQAE (Software Quality Assessment Exersice), entre otros, donde se detalla la estructura y características de calidad presentes en cada uno de éstos.
Muchas de estos modelos han ido modificándose y evolucionando a través de los años, con el fin de establecer características de calidad y sugerir métricas para su medición, tal es el caso, del estándar internacional ISO/IEC 9126, referenciado por Wagner [5].
Estos modelos se abordan desde diferentes perspectivas y clasificaciones a través de la literatura. Para este caso particular, se consideran los modelos jerárquicos, basados en el meta-modelo e implícitos de calidad.
Los modelos jerárquicos utilizan la descomposición jerárquica definiendo características de calidad, teniendo por ejemplo, factores como fiabilidad y mantenibilidad. Entre los modelos jerárquicos más populares se pueden mencionar los descritos por Boehm, Dromey, McCall, FURPS y la norma ISO 9126, posteriormente, esta norma evolucionó, convirtiéndose en la norma ISO / IEC 25010 [5]. Estos modelos intentan evaluar la calidad del software en general, y no de una manera especializada o con un análisis evolutivo, por lo tanto, algunas subcaracterísticas de calidad se omiten o no son abordadas de manera explícita [6].
Por otra parte, los modelos basados en meta-modelos, se definen a partir del concepto de *propiedad, atributo e impacto*. Donde la propiedad, de manera generalizada incluye los atributos, cualidades o características del objeto. Así mismo, el atributo se emplea para caracterizar la entidad y finalmente, el impacto es el grado en el cual un atributo influye en otro [7]. De esta forma, este tipo de modelos dan pie a la relación que debe existir entre los criterios establecidos por los modelos jerárquicos y su relación con la forma en que cada uno será evaluado.
Un ejemplo de este tipo de modelos es COQUAMO (Constructive Quality Model), el cual se preocupa por la necesidad de precisión en la medición de la calidad, generando otros modelos a partir de éste, y haciendo una distinción entre las medidas de calidad objetivas y subjetivas [8].
Finalmente, los modelos de calidad estadísticos e implícitos, han sido propuestos para identificar propiedades de un producto, proceso u organización, y estimar de manera estadística sus factores de calidad, empleando para ello datos y patrones para realizar predicciones, como el comportamiento de falla de un producto de software a través del tiempo o patrones de error. Algunas herramientas son: FindBugs, Gendarme o PC-Lint, que los utilizan para la clasificación de tipos de problemas identificados y su impacto [5].

3. Visión General del modelo TD-CCS

3.1. Generalidades del modelo

El modelo planteado puede ser implementado por cualquier organización en la cual no se tenga ningún modelo de control de calidad, sin embargo, su aplicación y ciertas características específicas se enfocan para el desarrollo de las aplicaciones dentro de la plataforma e-learning .LRN, algunas de estas características se relacionan con la estructura de carpetas, portal de proyecto, entre otros que se detallan más adelante.
Para el modelo se establece una serie de acciones en procura de la calidad del software desarrollado, para evitar que los errores lleguen al usuario final, incorporando características de los meta-modelos, por medio de requisitos de calidad y cómo serán evaluados sus elementos.
Según Kläs, et al., citado por Nandakumar et al. [9] para la predicción de defectos y aseguramiento de la calidad, se debe combinar la opinión de los expertos y la historia de los proyectos en la organización, llevando a cabo actividades de calidad y mitigando el riesgo relacionado con la calidad de software en el producto terminado.
El modelo propuesto basa su estructura en la aplicación de una plantilla de calidad, estructurada por secciones de revisión para determinar si una apliación cumple o no con las características que se definieron para cada una, de igual forma, estas secciones se convierten en pasos de revisión dentro del proceso de control de calidad.
Los pasos asociados a este modelo dan inicio cuando el equipo desarrollador hace solicitud de la plantilla de control de calidad y completa el plan de pruebas. En la figura 1 se muestran los pasos del modelo.
El modelo es un punto de partida hacia el aseguramiento de calidad integral, considerando que la calidad del software no es una propiedad establecida o universal, sino que depende de cada proyecto o construcción en específico, para la definición de la misma [10].
A continuación, se detallan los pasos propuestos por el modelo TD-CCS.

3.1.1. Paso 1: Revisión de documentación. Consiste en la verificación de aspectos de estructura general de información referente a cada aplicación desarrollada, para dicho efecto en la organización se debe contar con una estructura mínima requerida y establecer aspectos de internacionalización según los idiomas requeridos.

3.1.2. Paso 2: Revisión de calidad de paquetes. Identificar los componentes relacionados con la distribución de carpetas según estándar asociado al ambiente de desarrollo de la organización. Adicional a la estructura del paquete de desarrollo, se verifican aspectos de calidad de código

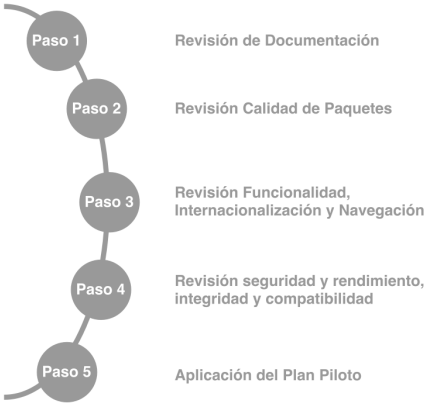

Figura 1. Diagrama de modelo de Control de Calidad del Software

y dependencias del paquete. Más adelante se detallan las características que se definieron en la aplicación del modelo.

3.1.3. Paso 3: Revisión de funcionalidad, internacionalización y navegación. Ejecutar las funcionalidades descritas en el plan de pruebas y monitorear el comportamiento de la aplicación y resultados obtenidos, así como aspectos de internacionalización que se hayan definido para la aplicación. Contempla aspectos relacionados con la interfaz gráfica web de la aplicación, con el fin de que las mismas cumplan con la calidad y estándar gráfico y visual que cada organización haya definido para sus aplicaciones. Es requerido contar con un estándar básico de diseño para la evaluación de este punto. Finalmente, se lleva a cabo pruebas de datos para validar la completitud, consistencia y exactitud de los datos.

3.1.4. Paso 4: Revisión de seguridad y rendimiento, integridad y compatibilidad. Identificar aspectos de seguridad en general para acceso a base de datos y acciones según permisos por rol y usuario, así como respuesta de la aplicación ante peticiones al servidor, verificando la consistencia de la información tras las acciones ejecutadas. Durante este paso se incluyen pruebas de compatibilidad en diferentes navegadores y sistemas operativos.

3.1.5. Paso 5: Aplicación del plan piloto. Verificar la consistencia y comportamiento esperado de la aplicación en un ambiente de producción, bajo condiciones normales de trabajo llevadas a cabo por un grupo de usuarios expertos, que permitirán identificar cualquier mejora para la posterior liberación del producto al resto de la organización.

3.2. Gestión de hallazgos

A través de la ejecución de los pasos descritos en la sección 3.1, el modelo propone completar una plantilla con los resultados obtenidos para cada sección. De igual forma, se procede a completar una sección de *hallazgos*, la cual se ha clasificado en diferentes categorías, siendo estas:

1) Error: son hallazgos que producen un error en el funcionamiento de la aplicación, tienen prioridad de resolución y de la corrección de éstos, depende la liberación del producto.
2) Mejora: son hallazgos que representan una oportunidad de mejora en beneficio de la aplicación. No representan un error, por lo cual, no afectan la liberación del producto.
3) Duda: se da cuando no se comprende una funcionalidad, comportamiento o anotación del plan de pruebas. No representan errores.

Adicionalmente, cada hallazgo debe ser priorizado, independientemente de su categoría. Las prioridades son: alta, media y baja, y queda a criterio del gestor de calidad su asignación, dependiendo la medida en que afectan el funcionamiento de la aplicación.
Una vez que la plantilla de calidad ha sido completada por el gestor de calidad, se debe registrar todos los hallazgos, si los hubiera, y comunicar la finalización del proceso de pruebas al equipo de desarrollo.
A partir de este momento, se inicia un proceso iterativo entre correcciones y mejoras efectuadas, hasta que todos los hallazgos sean gestionados.

4. Aplicación del modelo en el TEC Digital

A partir de las generalidades del modelo, se procede a su aplicación en el TEC Digital, donde los proyectos de desarrollo de software son orientados al e-learning, empleando para ello la plataforma .LRN.
Este gestor de aprendizaje es una aplicación open-source que colabora en la creación de comunidades de aprendizaje, adicionalmente, .LRN se encuentra soportado por el Framework de desarrollo web OpenACS y distribuido bajo la licencia pública GNU (GLP) [11].
De forma detallada y para cada paso del modelo, se muestran los criterios y descripción del comportamiento asociado e incluidos dentro de la plantilla de control de calidad. La matriz generada da pie al instrumento de control de calidad y de trazabilidad de las aplicaciones revisadas.
A continuación, se detalla la aplicación de cada paso del modelo.

4.1. Paso 1: Revisión de documentación

Se efectúa una revisión de la documentación asociada a la aplicación y que se constituye en una carta de presentación de cada proyecto, permitiendo a cualquier usuario el poder comprender su funcionalidad y facilitar la transferencia de conocimientos. Se tiene la ventaja que dentro del ambiente de .LRN se cuenta con la aplicación de portales, por lo cual, la documentación es generada dentro del mismo sitio donde se almacenan las aplicaciones.
En el cuadro 1 se detalla cada criterio de calidad considerado para la revisión del portal de acuerdo a los criterios definidos en la organización.

- *Métrica*: cumplimiento de cada criterio establecido a nivel de portal de acuerdo con su comportamiento

Criterio	Descripción comportamiento esperado
Estructura de pestañas	Según estructura definida por TEC Digital, debe contener las pestañas: Presentación General, Arquitectura de sistema, Documentos, Infográfico y Admin
Descripción	Debe contener descripción de la aplicación
Video Demo	Cuenta con video demostrativo
Justificación	Se incorpora justificación del proyecto
Obj. General	Cuenta con objetivo general
Obj. Específicos	Cuenta con objetivos específicos
Estado Proyecto	Se incluye estado del proyecto actualizado, con detalle de fechas por etapa
Última actualización	Se incluye la última fecha de actualización del portal, y detalle del cambio
Involucrados	Se incluyen involucrados en el proyecto
Modelo Conceptual	Se incluye diagrama del modelo conceptual
Arquitectura Sistema	Se incluye diagrama de arquitectura
Diagrama BD	Se incluye diagrama de Base de Datos
Requerimientos	Especificación de requerimientos que debe cumplir la aplicación
Documentos	Debe contener la documentación creada en el ciclo de vida del proyecto
Admin	Permite administrar el portal del proyecto, se muestra por definición cuando se crea el portal, visible para usuarios administradores

Cuadro 1. MATRIZ DE CALIDAD DEL PORTAL

esperado o característica asociada.
Para estos criterios se manejan valores binarios, es decir se indica *"No"*, en caso que no haya cumplimiento del criterio, o *"Si"* en caso de cumplir el criterio, en este último caso, se indica si el mismo se encuentra en español e inglés -idiomas obligatorios para la información del portal-. Adicionalmente, se brinda un espacio para observaciones. En caso de no cumplir con algún criterio, el responsable del proyecto debe incorporar las correcciones.

4.2. Paso 2: Revisión de calidad de paquetes

Verifica el cumplimiento del estándar de contenido de los paquetes de la aplicación. Según la estructura básica de un paquete en .LRN / OpenACS [12], se cuenta con tres carpetas: "sql", "tcl" y "www", así como un archivo de información del paquete "paquete.info", esta estructura responde a su patrón de arquitectura *Modelo Vista Controlador (MVC)* [13].
Mediante la carpeta "catalog" se incluye el catálogo de internacionalización, y eventualmente, se puede hacer uso de la carpeta "lib" con el propósito de extender el paquete usando otras librerías. En la figura 2, se puede visualizar la estructura básica de un paquete .LRN y sus componentes.
El cuadro 2 detalla cada uno de los criterios relacionados con la calidad del paquete o código fuente del proyecto.

- *Métrica*: se emplea una valoración binaria, para indicar si *"Lo aplica"* o *"No lo aplica"*, adicionalmente, se ofrece un espacio para observaciones, con el fin de ampliar cualquier información de importancia. Esto para cada uno de los criterios señalados.

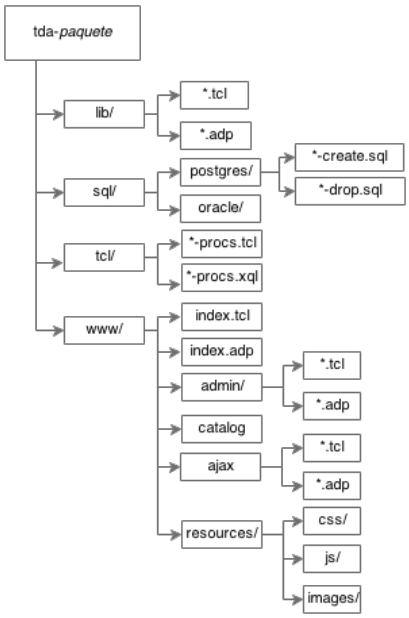

Figura 2. Estructura básica de un paquete en .LRN. Elaboración propia basada en Hernández et al. [12].

4.3. Paso 3: Revisión de funcionalidad, internacionalización y navegación

Este paso fue uno de los principales cambios de impacto con el modelo de control de calidad de software, por cuanto las pruebas dejaron de ser realizadas por los desarrolladores para ser aplicadas por el personal a cargo de la función de control de calidad y desde la perspectiva del usuario final.

4.3.1. Funcionalidad. Se realiza paso a paso cada actividad descrita por el desarrollador en el plan de pruebas y para cada actividad se indica el resultado obtenido. Este paso tiene el fin de observar la secuencia de acciones y respuesta de la aplicación, así como parametrizaciones requeridas.
Adicional a las actividades contempladas por el desarrollador, se efectúa cualquier tipo de interacción para poner a prueba la aplicación, así como la emulación de situaciones que puedan conducir a error.
Durante este paso el gestor de calidad asume un rol de usuario final, por lo cual, muchos de los hallazgos de tipo *mejora* y *duda* surgen a partir de estas acciones.

- *Métrica:* para cada actividad ejecutada y dependiendo del resultado obtenido, se indica si es correcto o incorrecto. Se puede indicar comentarios de aclaración para cada actividad.

4.3.2. Internacionalización. Se realiza una verificación para garantizar que las aplicaciones cuentan con los catálogos de internacionalización suministrados por .LRN, y según los idiomas establecidos por la organización.
A nivel de TEC Digital la información de la aplicación debe estar en idiomas español e inglés, a excepción de la información que proviene de bases de datos y web services.

Criterio	Descripción comportamiento esperado
Distribución carpetas	Debe cumplir con la estructura de carpetas: catalog, sql, tcl, www, .info, lib (opcional)
.info	Indicar dependencias, callbacks, versión
Nombre significativo	Consistencia de nombres en paquete y archivos
Archivos /www	Archivos .tcl y .adp con documentación interna, autores y fechas
Responsable	En archivo .info se debe señalar al menos un responsable de la aplicación
Internacionalización	Cada aplicación debe tener su correspondiente internacionalización
Esquema BD	Debe cumplir con el estándar sch_nombre
Querys en .xql	Querys empleados deben estar en el archivo .xql
xx-create.sql	Contiene archivo de creación -puede o no tenerlo-
xx.drop.sql	Contiene archivo de borrado -puede o no tenerlo-
Archivos -procs	Carpeta /tcl: contiene archivos .tcl y .xql
xx-procs.tcl: procedimientos	En este archivo se definen los procedimientos del paquete
Uso de namespace's	Alusivo al nombre de la aplicación, debería ser igual al nombre del paquete
/www interfaz gráfica	No debe haber lógica de programación
Consistencia nombre	En carpeta /www por cada archivo .tcl y .adp, debe existir un .js y .css con el mismo nombre
Código js en .js	No debe haber código js fuera de un archivo js
Código css en .css	No debe haber código de estilos fuera de un archivo css
/www/recursos	Uso adecuado para imágenes, gráficos y recursos similares

Cuadro 2. MATRIZ DE CALIDAD DEL PAQUETE

- *Métricas*: no cuenta con una métrica definida, los hallazgos en esta sección son incluidos dentro de la plantilla de revisión en el apartado de *hallazgos* y su corrección es de carácter obligatorio para el equipo desarrollador.

4.3.3. Navegación. En TEC Digital se cuenta con un área de Comunicación Visual, encargada, entre otros aspectos del desarrollo de interfaces gráficas centrada en la usabilidad en función a las necesidades del usuario, para lo cual se creó un estándar de interfaces.
En este paso, se realiza una comparación entre la interfaz gráfica web implementada y el diseño propuesto, con el fin de que las aplicaciones cumplan con el estándar definido.
Así mismo, a nivel de usuario final se efectúa una revisión de la aplicación evaluando aspectos de interfaz, como el comportamiento esperado de botones y las acciones asociadas, manejo de eventos anómalos como fallo de servicios web, entre otros.
Como parte del estándar de desarrollo se cuenta con una librería llamada "tds-lib" basada en el framework Bootstrap para el CSS y componentes de interfaz web con funcionalidades en JavaScript, lo que permite el diseño y desarrollo de páginas web con técnicas "responsive web design" [14]. Esta librería es de uso obligatorio para el equipo de desarrollo en procura de la reutilización y simplificación de componentes, y estandarización de elementos gráficos y visuales.

Criterio	Descripción comportamiento esperado
Ajuste al diseño	Se ajusta al diseño realizado en comunicación visual
Componentes - spinner	Debe controlar máximo, mínimo, no debe permitir letras
Componentes - textarea	Debe controlar máximo, mínimo, letras o Cantidades -según corresponda-. Debe permitir tildes y caracteres especiales
Componentes - textfield	Debe controlar máximo, mínimo, letras o cantidades -según corresponda-. Debe permitir tildes y caracteres especiales
Listas de despliegue	Muestra datos relacionados con parámetros ingresados. Si no hay elementos no debe mostrar ningún dato. Debe seguir un orden (alfabético, por código, otro)
Botones	Se muestran botones habilitados y deshabilitados correspondientes a la aplicación. Efectúan la función y eventos asociados.
Tabs	Se muestran los tabs correspondientes a la aplicación. Funcionan correctamente
Checkbox	Muestra checkbox propios de la aplicación. Se puede seleccionar varios valores
Radio Buttons	Se muestran los radio buttons que corresponden a la aplicación. Se puede seleccionar sólo uno de los valores desplegados
Tooltips	Muestra los comentarios correspondientes
Data Grids	Funcionan correctamente. Muestra datos
Mensajes de alerta	Muestra los mensajes de alerta respectivos
Navegación en pantalla	Se puede acceder a todos los componentes

Cuadro 3. MATRIZ DE CALIDAD DE NAVEGACIÓN

En el cuadro 3 se detallan los criterios de calidad referentes a la navegación y la descripción del comportamiento esperado.

- *Métricas*: cantidad de errores o cantidad de diferencias entre el diseño final presentado por la aplicación y la especificación dada por Comunicación Visual. También, se mide según la cantidad de errores con respecto al comportamiento de cada componente de navegación definido en los criterios del cuadro 3. Se empleó una escala de 1 a 5, donde 5 es la máxima calificación en caso de no encontrarse errores o diferencias. Sin embargo, al no tenerse un rango definido, tiende a ser subjetiva y se ubica como punto de mejora.

4.4. Paso 4: Revisión de seguridad y rendimiento, integridad y compatibilidad

4.4.1. Seguridad y rendimiento. En esta etapa se debe verificar aspectos de inicio de sesión, así como permisos o visualizaciones a los que se tiene acceso según el perfil de usuario con el cual se efectúan las pruebas, para evitar el acceso por usuarios no autorizados.
Contempla todas aquellas pruebas de inyección de sql a la base de datos por la interacción del usuario y las aplicaciones, garantizando la integridad de la información.
En complemento, el modelo propone efectuar pruebas de carga de datos y tiempos de respuesta.
En el cuadro 4 se detalla cada uno de los criterios de calidad

Criterio	Descripción comportamiento esperado
Datos sensibles por URL (seguridad)	Envío de datos por POST o algún método seguro
SQL injection (seguridad)	Comprobar la seguridad de campos de texto ante inyección SQL
Inicio de sesión (seguridad)	Probar entrar sin iniciar sesión. Validar acceso a información según tipo de usuario
Pruebas de estrés (rendimiento)	Comportamiento de la aplicación con varios usuarios accediendo
Pruebas de carga (rendimiento)	Carga adecuada de datos ante acción efectuada

Cuadro 4. MATRIZ DE CALIDAD DE SEGURIDAD

considerados, así como su respectivo comportamiento esperado. Los criterios se subdividen en aspectos de seguridad y aspectos de rendimiento.
Para algunas aplicaciones específicas se realizan pruebas reales con múltiples usuarios, para lo cual se prepara un ambiente de pruebas y se analiza el comportamiento a nivel de registro de información, tiempo de respuesta de la aplicación y del mismo servidor, monitorizando todo el proceso de la prueba. Estos datos son trasladados al equipo desarrollador para incorporar mejoras o correcciones.

- *Métricas*: cantidad de hallazgos o errores en cuanto a comportamiento esperado de cada criterio definido en el cuadro 4.
  Se utiliza un puntaje de entre 1 y 3 puntos, donde el 3 implica el cumplimiento total del criterio y 1 su incumplimiento. Estos rangos se establecieron durante la definición inicial de la plantilla, sin embargo, se ha visto que son subjetivos, por lo cual se considera dentro de los trabajos a futuro.

4.4.2. Integridad. Para valorar la integridad de la aplicación el modelo plantea llevar a cabo pruebas de datos, a nivel práctico, este tipo de pruebas se llevan a cabo mediante la inserción de caracteres o datos incorrectos en puntos específicos de la aplicación, con la finalidad de garantizar la integridad y correctitud de las transacciones, así como el manejo de este tipo de eventos.
De igual forma, se realizan pruebas para asegurar la integridad de la información a nivel de base de datos, para este efecto, se prueba realizar inserciones, borrados y actualizaciones desde la aplicación, para corroborar que se mantenga esta característica de calidad a lo largo de las diferentes interacciones del sistema y el usuario.
El cuadro 5 detalla cada uno de los criterios de calidad y su comportamiento esperado.

- *Métrica:* cantidad de hallazgos o errores en cuanto a comportamiento esperado. Al igual que en el punto anterior, la evaluación se definió en una métrica de 1 a 5 puntos por rubro, donde el 5 representa la ausencia de errores o hallazgos, sin embargo, fue una definición inicial de la plantilla de revisión, la cual es subjetiva por lo cual se considera como mejora dentro de los trabajos a futuro.

Criterio	Descripción comportamiento esperado
Correctitud de transacciones e información	Las inserciones, actualizaciones y borrados, mantienen la información consistente de acuerdo a lo requerido
Datos no válidos o requeridos	Inserción de datos no válidos en los diferentes componentes del sistema. Validación en caso que no se ingresen datos requeridos
Manejo de eventos anómalo	Ante eventos anómalos mantiene la integridad de la información, se informa al usuario
Integridad referencial	Mantiene relación entre datos durante la ejecución del sistema, BD refleja cambios
Soporte a paralelismo	Integridad de la información aún cuando diferentes fuentes la acceden y actualizan

Cuadro 5. MATRIZ DE CALIDAD DE INTEGRIDAD

4.4.3. Compatibilidad. En este paso se lleva a cabo una verificación de compatibilidad multi-plataforma y multi-navegador con la aplicación. La verificación consiste en realizar pruebas de usuario para los sistemas operativos y navegadores web del cuadro 6 y cuadro 7.

Sistema Operativo	Versión
Linux	Debian 6+, Ubuntu LTS
Windows	Windows 7 y superiores
Mac OS	10.7+

Cuadro 6. COMPATIBILIDAD SISTEMA OPERATIVO

Navegadores Web	Versión
Chrome	última versión -1 o última versión
Mozilla Firefox	última versión -1 o última versión
Internet Explorer	6+
Safari	5.1+

Cuadro 7. COMPATIBILIDAD NAVEGADORES WEB

De acuerdo con el modelo propuesto, se realizan revisiones en las plataformas y navegadores establecidos, sin embargo, se excluyen pruebas en versiones anteriores a las indicadas y dispositivos móviles. Cabe destacar que no se cuenta con registro de pruebas para el browser de Microsoft Edge ni Opera. En el cuadro 8 se detallan los criterios de calidad y su comportamiento esperado.

- *Métricas:* trabaja con la cantidad de hallazgos o errores en cuanto a soporte del navegador para cumplir con el comportamiento esperado.
  Se trabaja bajo una escala de 1 a 5, siendo 5 la máxima puntuación, es decir el comportamiento esperado, sin embargo, y al igual que en puntos anteriores, se convierte en una evaluación subjetiva.

Criterio	Descripción comportamiento esperado
Mozilla Firefox, Chrome, Internet Explorer, Safari	El sistema puede ser visto y ejecutado en este navegador de manera correcta
Linux, Windows, Mac OS	El sistema puede ser ejecutado en el sistema operativo
Interconexión e integración acordada	Interconexión con otros paquetes según acuerdos y aún en condiciones de error

Cuadro 8. MATRIZ DE CALIDAD DE COMPATIBILIDAD E INTEGRACIÓN

4.5. Paso 5: Aplicación del plan piloto

Se propone aplicar un plan piloto orientado al usuario final, se sugiere trabajar con un grupo de usuarios que hayan sido considerados para la toma de requerimientos.
Se hace uso de la experiencia del usuario final, para agregar valor a la aplicación y capturar errores o comportamientos inesperados del producto de software. Se busca generar mayor confianza y credibilidad hacia la aplicación, generando menor resistencia y mayor aceptación.
En dicha etapa es imperativa una comunicación eficiente y oportuna, al trabajar en conjunto con usuarios expertos y principales interesados en el uso de la aplicación.
Otro aporte de este paso, consiste en probar la correspondencia de manuales o infografías para que sean una guía autosuficiente para el uso de la aplicación por parte de cualquier usuario.

5. Evaluación y resultados obtenidos

Con la metodología de desarrollo actual se han llevado a cabo varios proyectos y la plantilla propuesta dentro de la metodología TD-CCS ha sido aplicada a veintiseis proyectos de software de TEC Digital. Para algunos proyectos no se aplicó en su totalidad, sino de manera parcial, enfocado a la verificación de funcionalidad de la aplicación.
Se analizaron los datos de la plantilla de control de calidad para diez proyectos, los cuales actualmente se encuentran en ambiente de producción. Dichos proyectos estuvieron en revisión de control de calidad durante los años 2014 y 2016, y el análisis de datos se efectuó en dos etapas.
La primer etapa incorporó los diez proyectos seleccionados y considerando que el enfoque de este modelo es la calidad para el usuario final, el análisis de la información se centra en aspectos de funcionalidad, haciendo énfasis en el tipo de hallazgos, así como la cantidad de tareas revisadas correctas e incorrectas, en comparación con la totalidad de tareas indicadas en el plan de pruebas de cada aplicación.
En la segunda etapa, se consideró cuatro aplicaciones para aspectos de navegación, seguridad e integridad, y cinco aplicaciones respecto a compatibilidad, esto por cuanto se incluyeron datos más específicos en cada característica.
El análisis no incluyó la categoría *Documentación*, *Calidad de Paquetes* ni *Plan Piloto*. Se detallan los resultados:
Primer etapa:

1) Un 100 % de los proyectos revisados presentaron al menos dos hallazgos.
2) De 389 tareas revisadas e incluidas en el plan de pruebas, un 76,61 % (298 tareas) fueron categorizadas como correctas.
3) El 23,39 % (91 actividades) se clasificaron como incorrectas por comportamiento contrario a lo esperado según el plan de pruebas de cada aplicación.
4) Adicional a las actividades del plan de pruebas, se realizaron pruebas integrales de funcionalidad, y se identificó 454 hallazgos adicionales, que fueron reportados y gestionados por el equipo de trabajo del proyecto, evitando que llegaran al usuario final.
5) La mayoría de hallazgos identificados en las aplicaciones analizadas son de tipo error un 72,03 % (327 hallazgos), de tipo mejora se identificó un 21,81 % (99 hallazgos) y tipo duda 6,17 % (28 hallazgos), representado en la figura 3.
6) Se priorizó los 327 hallazgos de tipo error, obteniendo: 72,78 % de prioridad alta (238 hallazgos), de prioridad media 14,37 % (47 hallazgos) y de prioridad baja 12,84 % (42 hallazgos).
7) Con respecto a los 99 hallazgos de tipo mejora, se tiene: 55,56 % de prioridad alta (55 mejoras), de prioridad media 36,36 % (36 hallazgos) y 8,08 % con prioridad baja (8 hallazgos).
8) Finalmente, para los 28 hallazgos clasificados como duda, se tiene: 89,29 % con prioridad alta (25 hallazgos), 7,14 % de prioridad media (2 hallazgos) y 3,57 % con prioridad baja (1 hallazgo).

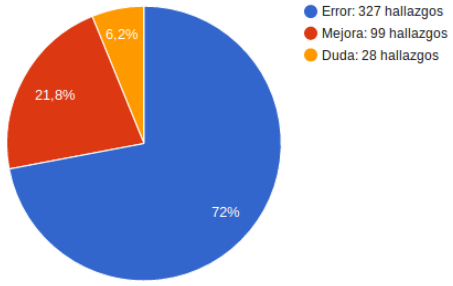

Figura 3. Tipo de hallazgos

Segunda etapa: Se consideró la totalidad de características evaluadas por proyecto y se promedia con las características que no cumplen con el comportamiento esperado.

1) En cuanto a aspectos de navegación, se consideró la información de cuatro aplicaciones. En promedio se obtiene un 41,67 % de características de navegación que no se cumplen de los aspectos evaluados.
2) En aspectos de seguridad, se registró un 11,11 % de incumplimiento de los elementos revisados. La recopilación se obtuvo de un total de 4 aplicaciones.
3) Para efectos de análisis en cuanto a integridad se consideró 4 aplicaciones y se obtuvo un 54,17 % de cumplimiento de los elementos considerados.
4) En cuanto a elementos de compatibilidad se registró un promedio de 6,45 % de elementos que no cumplen con las características definidas en este aspecto, según análisis de cinco aplicaciones.
5) La figura 4 muestra el promedio de elementos que se cumplen o no en las revisiones en materia de: navegación (cumple un 58,3 % de elementos), seguridad y rendimiento (cumple un 88,9 %), integridad (cumple un 54,2 % de elementos) y compatibilidad (cumple 93,5 % de elementos).

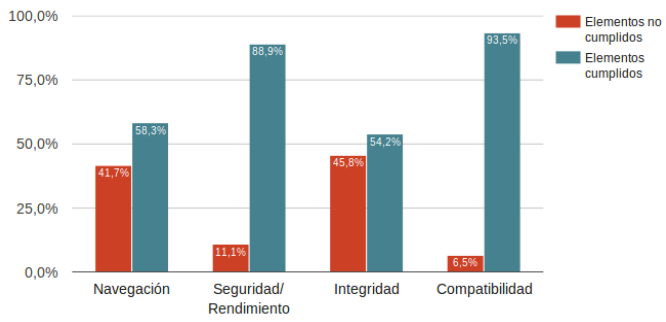

Figura 4. Evaluación de elementos en Navegación, Seguridad y Rendimiento, Integridad y Compatibilidad

6. Limitaciones

1) No todos los proyectos revisados fueron evaluados en las mismas características de calidad, por cuanto no existe una estandarización en las revisiones.
2) La aplicación del modelo se efectuó parcialmente sobre los proyectos, por cuanto el puesto de gestor de calidad ha experimentado mucha movilidad y carga de trabajo, por lo cual algunos pasos del modelo dejaron de registrarse, y en otros casos el manejo de hallazgos históricos no se conservó, sino que se modificó al ser solventada la observación.
3) Muchos de los aspectos evaluados se prestan para ser subjetivos e interpretados de acuerdo con el criterio propio del gestor de calidad.
4) La plantilla como instrumento de control y seguimiento ha ido experimentando cambios, por lo cual, algunas características de calidad han variado en cuanto a su definición y aplicación.
5) Se debe incorporar la evaluación de características asociadas al aseguramiento de la calidad en materia de e-learning específicamente.

7. Conclusiones y trabajo futuro

El modelo TD-CCS propuesto está orientado a proyectos de desarrollo de software para una organización de e-learning, en la cual el control de calidad se ejecuta en etapas tardías del ciclo de vida del software.
El instrumento permite una inspección manual de la totalidad de la aplicación y llevar a cabo la función de medición, para establecer un comparador entre el comportamiento esperado y el real, que aunque en algunos casos es subjetivo, es un punto de partida.
El modelo busca ser una guía estandarizada, teniendo en cuenta la tendencia a la mejora continua del instrumento a la luz de las mejores prácticas en materia de calidad del software y madurez del modelo en la organización.
Aunque con la metodología propuesta se logró identificar y gestionar los hallazgos, para que éstos no fueran trasladados al usuario final, afectando su aprendizaje en el proceso de interacción con las herramientas de software, es importante que esta gestión se realice desde etapas tempranas del proyecto, para reducir el retrabajo y tiempo de corrección.
El proceso permitió identificar que muchos errores no fueron identificados por los equipos desarrolladores, resaltando el impacto de las actividades de calidad.
Se debe establecer mecanismos para documentar la experiencia generada durante la aplicación de plan piloto.
De igual forma, se debe establecer métricas objetivas para cada uno de los criterios que se evalúan por medio de la plantilla de calidad, de manera que la labor de medición y auditoría sea más completa y sencilla.
Así mismo, es importante llevar a cabo esfuerzos para medir y determinar el impacto que la calidad conlleva en el componente de enseñanza y aprendizaje.
Finalmente, se debe agregar revisiones de accesibilidad y ejercer un monitoreo constante de las aplicaciones en ambiente de producción, para identificar cualquier hallazgo que pueda surgir durante su uso final. Así mismo, reforzar las acciones en materia de seguridad y rendimiento.

Referencias

[1] R. Pressman, "Ingeniería del software-enfoque práctico mc graw hill 5ª," *Edición Año*, 2002.

[2] J. Espinoza and M. Chacón, "Tec digital: Una iniciativa de implementación de e-learning en costa rica," in *XVIII Congreso Iberoamericano de Educación Superior en Computación*, 2010.

[3] S. Pressman Roger, "Ingeniería de software un enfoque práctico (sétima edición ed.)," 2010.

[4] J. J. Moreno, L. P. Bolaños, and M. A. Navia, "Exploración de modelos y estándares de calidad para el producto software," *Revista UIS Ingenierías*, vol. 9, no. 1, 2010.

[5] S. Wagner, *Software product quality control*. Springer, 2013.

[6] H. P. Breivold and I. Crnkovic, "Analysis of software evolvability in quality models," in *Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on*. IEEE, 2009, pp. 279–282.

[7] K. Lochmann and A. Goeb, "A unifying model for software quality," in *Proceedings of the 8th international workshop on Software quality*. ACM, 2011, pp. 3–10.

[8] E. Duggan, *Measuring information systems delivery quality*. IGI Global, 2006.

[9] R. Nandakumar, A. Lal, and R. Parmar, "State of the art in software quality assurance," *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 3, pp. 1–6, 2014.

[10] S. Wagner, S. Chulani, and B. Wong, "8th international workshop on software quality (wosq)," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM, 2011, pp. 524–525.

[11] J. A. Garita, I. Alpizar-Chacon, and M. Chacón Rivas, "Openacs/dotlrn integration with itcr platform," in *8th OpenACS/dotLRN Conference*, 2009.

[12] R. Hernández, A. Grumet, and O. C. Team, "Openacs: robust web development framework," in *Tcl/Tk 2005 Conference, Portland, Oregon*, 2005.

[13] F. Sotelo Gómez, A. Ordóñez, and M. F. Solarte, "Marco de referencia para la integración de recursos web como servicios de e-learning en. lrn," *tecnura*, vol. 19, no. 46, pp. 79–91, 2015.

[14] E. Harb, P. Kapellari, S. Luong, and N. Spot, "Responsive web design," 2011.