Manual para el control de calidad

" La función de un buen software es hacer que lo complejo aparente ser simple "

Grady Booch

Universidad Tecnológica de San Luis Río Colorado
carrera de Ingeniería de
Tecnologías de la Información y Comunicación

Dirección: Ave. Jalisco Y calle 59 s/n
Col. Progreso C.P. 83458
San Luis Río Colorado, Sonora, México.
Teléfono: (653) 518 5146 Ext. 123
Correo: tics@utslrc.edu.mx

Autores: Martin Galvan Covarrubias

Asesores: MEEE. Susy Mercado Avilés
LSCA. Ricardo Alejandro Soto Morales
LSCA. Arnoldo Delgado González

Agosto del 2019

Índice

Tabla de ilustraciones

1. Concepto de calidad

Antes de iniciar por completo con el manual es necesario conocer información relevante del tema con el objetivo de entrar en contexto poco a poco. Para comenzar, es necesario definir el concepto de calidad, Roger S. Pressman (2010) define la calidad de software de la siguiente manera “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.”

La calidad es un factor de mucha importancia ya que de esta depende que los resultados que se obtengan cumplan con lo que es solicitado por el cliente. Además, al tener calidad en los procesos se evitan atrasos en los proyectos, los cuales en ocasiones terminan por salirse del presupuesto planeado al inicio, esto es debido a defectos que llevan al retrabajo de actividades.

La calidad de un sistema podría pensarse a simple vista que es responsabilidad solamente de los desarrolladores, pero esto no es así, para asegurar la calidad se debe de trabajar de manera coordinada con los desarrolladores, analistas, administradores y todos los involucrados en el proyecto.

Todas las actividades para alcanzar la calidad tienen como objetivo entregar un producto final con el cual el cliente se encuentre satisfecho, y que el resultado obtenido tenga un desempeño eficiente y confiable para dar ventajas sobre los competidores del mercado.

Alcanzar la calidad no es una tarea sencilla, existen dos factores muy importantes que se deben de tomar en cuenta que son los recursos y el tiempo que se tiene disponible para desarrollarlo, se debe de buscar un equilibrio entre la calidad de los resultados con el tiempo que va ser invertido para el desarrollo ya que con mayor tiempo mayores serán los costos.

A lo largo del progreso del manual se tocarán diferentes etapas del desarrollo de software, en estas etapas se utilizarán formatos que permitirán evaluar los resultados de cada etapa.

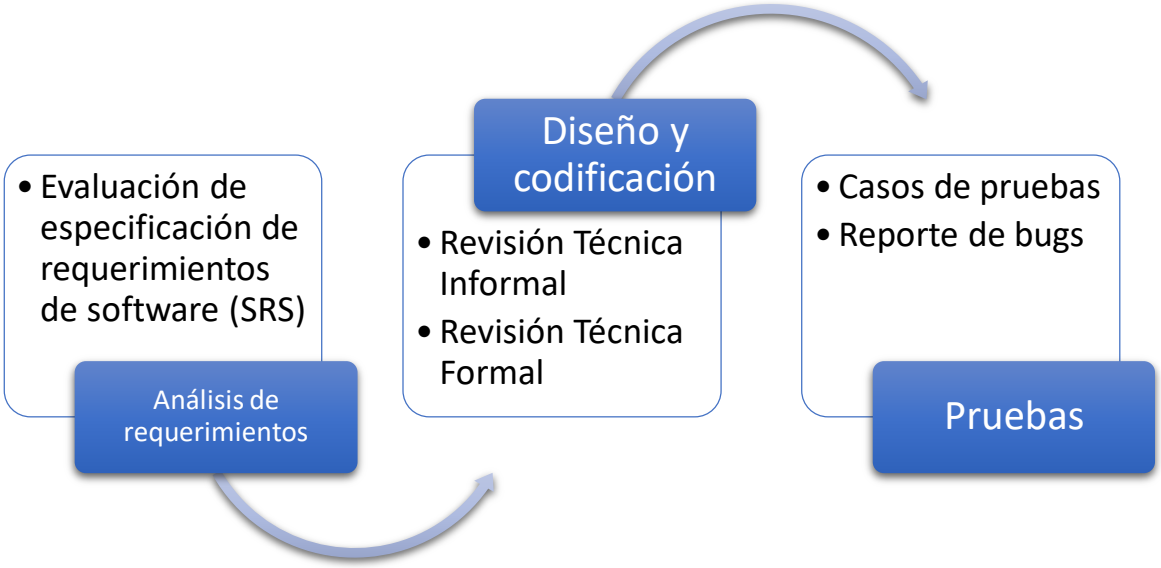

Ilustración 1 Etapas de desarrollo y formatos a utilizar

Para finalizar con el manual se desarrollará un apartado que tratará de las habilidades y actitudes con las que debe de contar un probador de software (también conocido como tester, por su denominación en inglés) para llevar a cabo un mejor desempeño en este rol.

2. Evaluación de requerimientos

La primera etapa a evaluar es el análisis de requerimiento donde se realiza un levantamiento de requerimientos, una vez completada esta actividad se procede a realizar un documento llamado Software Requirements Specitication **(**SRS) donde se incluye toda la información que debe de contener el sistema de software a desarrollar, durante esta actividad se especifican y definen los requerimientos que serán realizados durante el desarrollo del proyecto. Este documento debe de ser realizado tomando en cuenta el alcance del proyecto. Por esta razón, es una de las actividades con mayor importancia, el no contar con un análisis correcto y adecuado podría llevar al fracaso del proyecto.

Al completar el análisis de requerimientos será utilizada una lista de verificación o checklist que permitirá evaluar el documento desarrollado por los analistas, este formato consiste en una serie de preguntas o criterios que se aplican sobre los requerimientos de software, que son presentados de forma escrita.

Esta lista de verificación puede realizar preguntas como:

- ¿Se han especificado los requisitos de hardware y software?
- ¿Se han realizado consideraciones de seguridad?
- ¿Está escrito el requerimiento en un lenguaje claro y conciso?
- ¿El requerimiento es único? (no existe duplicidad con otro requerimiento).

La lista de verificación sirve de marco de trabajo y procedimental para revisar el requerimiento, facilitando su análisis de forma estructurada. Los requerimientos se pueden revisar sobre la definición del alcance.

El formato para este apartado se encuentra en él *Apéndice # 1 Evaluación de requerimientos*

3. Técnicas de revisión

Las revisiones permiten mejorar los resultados de las actividades que se realizan, estás pueden ser aplicadas en todas las etapas del proyecto, desde análisis hasta programación. Existen diferentes maneras de llevarlo para tener un control de las tareas que son realizadas. Estas se hacen con el objetivo de encontrar posibles mejoras o encontrar errores en las etapas iniciales, en otras palabras, las revisiones son un filtro.

Existen tres conceptos importantes que se deben de conocer en esta parte que son defecto, falla y error. La palabra defecto y falla pueden ser usadas como sinónimos y estas hacen referencia a un problema de calidad que es detectado hasta haberse liberado al usuario final. El término error hace referencia a los problemas de calidad detectados antes de entregar el software al usuario. A su vez el término bug será utilizado para hacer referencia a los conceptos anteriormente mencionados.

3.1 Revisión técnica informal

Es una revisión entre compañeros de trabajo, es una reunión casual, para esta no se requiere una planeación o una preparación previa por parte de los participantes. Una de las desventajas que se tiene es que no hay seguimiento a los errores que se encuentran. Este tipo de revisiones deben de ser tomadas como partes de las buenas prácticas de ingeniería de software. Los diseñadores deben de tomar notas de las observaciones que sean hechas por parte de los revisores para su posterior solución.

La manera de mejorar los resultados de estas revisiones es teniendo las siguientes preguntas en mente.

- ¿La distribución de los controles de la ventana está diseñada bajo el estándar establecido?
- ¿La interfaz de presentación necesita ser desplazada verticalmente?

• ¿Se usa de manera correcta el color, ubicación, tipografía y el tamaño de los controles?

• ¿Todas las opciones o funciones de navegación están representadas en el mismo nivel de abstracción?

• ¿Están etiquetadas con claridad todas las opciones de navegación de la interfaz?

3.2 Revisión técnica formal (RTF)

Buscan verificar que el software cumple con los requerimientos, no tiene errores, tiene un desarrollo uniforme de todo el sistema. Además, funciona como una capacitación ya que durante la revisión se pueden ver diferentes tipos de análisis, diseño y desarrollo. Para estas revisiones se deben de tomar en cuenta los siguientes puntos:

- Debe de haber de tres a cinco involucrados.
- Debe de existir una preparación previa por parte de todos los involucrados.
- La reunión es por lo general de dos horas.

Esta reunión debe de ser centrada en una parte específica y pequeña, en otras palabras, se revisan pequeños componentes del software en general, ya que con esto se reduce el alcance y se aumentan las probabilidades de encontrar errores.

¿Cuándo es momento de iniciar una revisión formal? El desarrollador debe de informar a su líder de proyecto que la actividad ha sido terminada para continuar con la revisión. El líder de proyecto debe de avisar al probador de software, este se encarga de notificar a las personas que van a estar involucradas para que se preparen, así mismo les hace llegar el producto que va a ser revisado. Los involucrados deben de revisar el producto, tomar notas y comprender lo que se está revisando. El líder del proyecto debe de revisarlo también y establecer una fecha de reunión. En esta reunión deben de estar los revisores, el desarrollador y el líder del proyecto.

Durante la reunión un revisor tomará el rol de secretario quién tomará registro de los acontecimientos más importantes de la revisión. El desarrollador debe de dar una introducción breve, posteriormente se procede a mostrar y explicar el trabajo realizado, mientras los revisores hacen sus comentarios con base a su preparación previa. Cuando se encuentran problemas válidos estos son anotados por el secretario. Al terminar la reunión todos los asistentes deben de decidir si

1) Aceptan el producto sin modificaciones.
2) Lo rechazan debido a errores graves (es necesaria otra revisión).
3) Se acepta de manera provisional con pequeños errores que serán modificados (no es necesario otra revisión).

El formato que debe de ser utilizado para la aplicación de estas revisiones se encuentra en el *Apéndice # 2 Formato para revisiones técnicas formales.*

Los campos a capturar en el formato son los siguientes:

- Proyecto: nombre del proyecto.
- Historial de revisiones
    - Fecha: fecha de la revisión.
    - Versión: versión de la revisión.
    - Descripción: detalles de la revisión.
    - Autor: persona que realizó la convocatoria de la RTF.
- Producto revisado
    - Nombre y versión del producto: se debe especificar el nombre, la versión y el área a la que corresponde el producto revisado.
    - Participantes de la revisión
        - Nombre: nombre completo.
        - Rol: rol que desempeña.
        - Iniciales: iniciales para usar como referencia en otras partes del documento.

        - Firma.
    - Técnica utilizada: documentos o formatos que fueron tomados como criterios. *5. Características a evaluar para mejorar la calidad de software.*
- Objetivos de la RTF: se plantean brevemente los aspectos del producto que serán revisados, qué propiedades de la calidad se buscará que cumpla y en qué grado, qué principios y estándares de calidad aplican al producto, si ya hubo revisión de otra versión del producto qué correcciones quedaron pendientes de realizar. Esta información es la misma que se incluyó en la convocatoria para la realización de la RTF enviada a los participantes.
- Problemas detectados
    - Problema: se describe el problema detectado estableciendo sus características, gravedad y especificando su ubicación.
    - Sugerencia de corrección: se sugieren las correcciones a realizar para que el producto cumpla con los principios y estándares de calidad establecidos y los procedimientos definidos.
- Evaluación
    - Estado actual del producto: se describe el estado actual del producto. por ejemplo, se debe rehacer o corregir, se puede entregar.
    - Acciones a tomar
        - Acción: se detallan las acciones a tomar para que el producto sea corregido y revisado en una nueva versión.
        - Responsable: iniciales de la persona encargada de dar solución.
- Fecha próxima revisión: se establece la próxima revisión de acuerdo a los puntos detallados anteriormente.

4. Estrategias de prueba para software

Existen muchas estrategias que pueden ser utilizadas para probar software. Anteriormente las pruebas eran realizadas hasta que el sistema estaba completamente construido con la finalidad de encontrar errores, esto daba como resultado software defectuoso. Actualmente se busca hacer revisiones con cada parte que se construye, primero se realizan pruebas a partes individuales del software, después se avanza con pruebas de integración de las unidades o módulos para culminar con evaluaciones al sistema completo construido.

4.1 Pruebas de unidad

Este tipo de pruebas se enfoca en la verificación de unidades o funcionalidades pequeñas, en un componente o en módulo del software. La interfaz del módulo se prueba para que la información fluya de manera correcta hacia y desde esta unidad. Se examina que los datos almacenados temporalmente mantienen su integridad durante los pasos de ejecución. Todas las posibles rutas se prueban para asegurar que todas las funciones se ejecutan correctamente por al menos una vez. Se prueban todas las restricciones que debe de tener. También deben de ser probadas las ventanas de errores.

Para estas pruebas se deben de diseñar casos de pruebas para descubrir errores provocados por cálculos erróneos, comparaciones incorrectas o flujo inadecuado. Los casos de pruebas deben de estar basados en el análisis de requerimientos. *4.3 ¿Qué son los casos de* pruebas?

El formato para casos de pruebas se encuentra en el *Apéndice #3 Formato para casos de prueba*.

Los campos a capturar en el formato son los siguientes:

- Proyecto: nombre del proyecto.
- Área: área, subproceso, módulo.

- Id: número consecutivo (folio).
- Caso de prueba: título descriptivo.
- Descripción: elementos, funcionalidades y acciones a realizar.
- Fecha: fecha de realización.
- Funcionalidad: por ejemplo, suscripción al servicio, consulta de órdenes.
- Requerimiento de ambiente de prueba: algún valor que sea necesario para ejecutar la prueba.
- Datos de entrada: valores ingresados y acciones.
- Datos esperados: ¿qué resultados se esperan?
- Datos obtenidos: ¿qué resultados dio?

4.2 Pruebas de integración

Una vez que todos los módulos se han probado individualmente es necesario iniciar con pruebas de todos en conjunto en otras palabras conectar todos los módulos, esto se hace con el objetivo de evitar problemas con los datos que son mostrados en las interfaces, también se busca revisar la estructura del programa que haya sido establecida durante el diseño.

Existen diferentes maneras de hacer estas pruebas, estas son algunas de ellas:

a) *Integración descendente:* en este tipo de pruebas se les asigna un orden jerárquico a los módulos. Las pruebas deben de iniciar en el módulo principal hacia los módulos más aislados, se podría decir que se inicia desde la raíz hasta llegar a las hojas.
b) *Integración ascendente:* también es necesario tener un orden jerárquico de los apartados o módulos, aquí se inicia desde los módulos aislados (nivel atómico o módulos que no dependen de otros) hacia el módulo principal, en otras palabras, se inicia desde las hojas hasta la raíz.

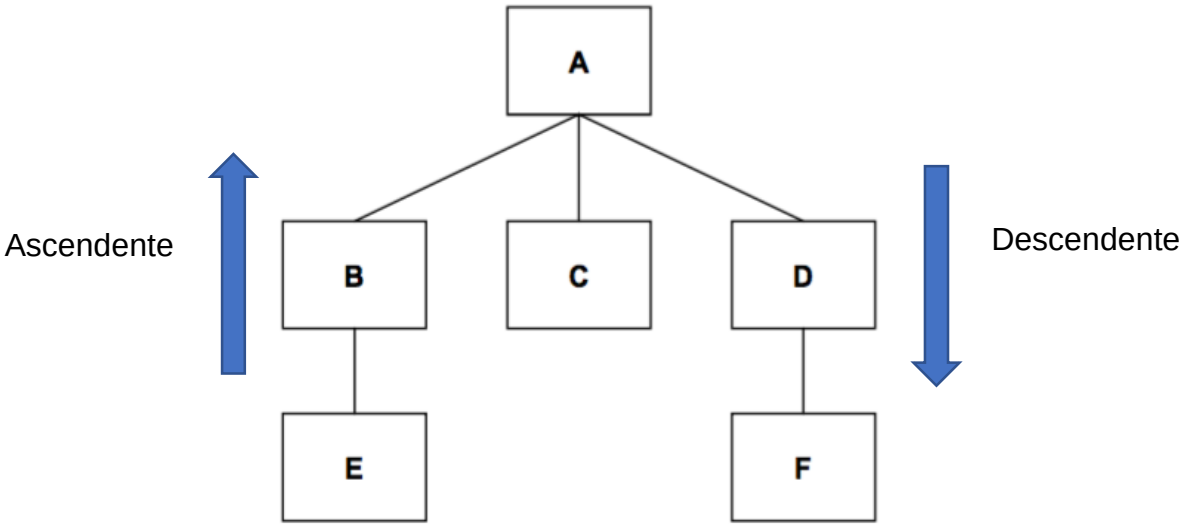

Ilustración 2 Maneras de realizar pruebas de integración

Para las pruebas de integración también puede ser utilizado el formato de casos de prueba que está en el *Apéndice #3 Formato para casos de prueba.* Al realizar estos casos de prueba se debe de tener en mente los puntos mencionados anteriormente.

4.3 ¿Qué son los casos de pruebas?

Los casos de pruebas son un conjunto de condiciones o variables bajo las cuales un probador de software determina si un sistema de software o una característica es parcial o completamente satisfactoria. En otras palabras, es un conjunto de pasos y resultados esperados que se crean a partir de los requerimientos de software.

Una vez completados los casos de pruebas se debe de hacer un informe de errores o bugs, estos bugs deben de ser reportados tan rápido como sea posible, ya que al entregar el reporte en un corto periodo de tiempo permitirá tener más tiempo para su reparación. Encontrar los bugs no es suficiente, estos deben de ser reportados y comunicados de una manera clara y eficiente por esta razón es importante conocer cuál es la manera correcta de llevarlos a cabo.

4.4 Categorías de los bugs

Es necesario clasificar los errores encontrados durante las pruebas realizadas esto permitirá que todos los involucrados se encuentren en el mismo canal. Los bugs pueden ser categorizados de la siguiente manera:

- Errores de seguridad: estos errores comúnmente envuelven el manejo incorrecto de la información transferida entre el usuario y la aplicación. Estos defectos son los que deben de tener la prioridad más alta. Por ejemplo, errores de autenticación y autorización a funciones específicas del sistema.
- Errores en la base de datos: manejo incorrecto de la información en la base de datos. Por ejemplo, tipos de datos incorrectos, tamaños de los campos, registros no borrados correctamente.
- Errores de funcionalidad: estos errores afectan la funcionalidad de la aplicación. Por ejemplo, errores de JavaScript, botones que no hacen la función correspondiente como guardar, borrar o agregar.
- Errores de interfaz de usuario: como el nombre lo menciona son problemas relacionados con UI (User Interface). Por ejemplo, errores ortográficos, alineamiento incorrecto, mensajes incorrectos.

4.5 ¿Cómo debe de ser un reporte de bugs?

Se deben de dar suficientes detalles al realizar el reporte teniendo en mente que las personas que lo van a leer no tienen conocimiento del bug. Esto significa que el reporte deberá estar escrito de una manera concisa, sencilla y clara.

El formato para los reportes de bugs se encuentra en *Apéndice # 4 Reporte de bugs.*

Los campos a capturar en el formato son los siguientes:

- Proyecto: nombre del proyecto.
- Área: área, subproceso, módulo.
- Fecha: fecha de realización.

- Id: número consecutivo (folio).
- Título: título del bug.
- Descripción: describir o explicar el bug encontrado.
- Categoría y prioridad: la prioridad puede ser manejada de la siguiente manera: muy baja, baja, media, alta y muy alta.
- Id del caso de prueba: identificador del caso de prueba con el cual se encontró el bug.
- Estado: los estados de los bugs se pueden manejar de la siguiente manera: abierto, pendiente, corregido, cerrado, reabierto.
- Pasos para reproducir: pasos necesarios para recrear el bug.
- Resultado real: resultado que muestra el sistema.
- Comentarios: comentarios del probador de software.

5. Características a evaluar para mejorar la calidad de software

Los siguientes formatos tienen como objetivo servir como herramientas para las Revisiones Técnicas Formales, estos documentos están basados en características que los sistemas deben de tener para mejorar la calidad de los resultados, enseguida se describe cada característica y a su vez se proporciona un ejemplo, es importante mencionar que estos formatos varían dependiendo el proyecto y los requerimientos del mismo por lo cual lo ideal sería que fueran adaptados a las necesidades del proyecto.

5.1 Mantenibilidad

Es la facilidad con la que un sistema o algún componente puede ser modificado para corregir fallos, mejorar su funcionamiento, atributos o adaptarse a cambios del entorno. Al tener una mayor mantenibilidad se reducen los costos para el mantenimiento. *Apéndice # 5 Evaluación de mantenibilidad*

5.2 Usabilidad

Es el esfuerzo necesario para aprender a operar el sistema, preparar los datos de entrada e interpretar los datos mostrados por el programa (datos de salida). *Apéndice # 6 Evaluación de usabilidad*

5.3 Portabilidad

Es el esfuerzo que se requiere para transferir el sistema de un entorno de hardware o software a otro entorno diferente. Se evalúa que el código fuente sea capaz de reutilizarse en vez de crearse nuevo código cuando el software pasa de una plataforma a otra. *Apéndice # 7 Evaluación de portabilidad*

5.4 Funcionalidad

Se refiere a la capacidad del software para proporcionar un conjunto de funciones que satisfagan las necesidades del usuario. Se evalúa el cumplimiento de los requerimientos, la exactitud de los resultados, la seguridad del producto y la interacción entre otros sistemas informáticos. *Apéndice #8 Evaluación de funcionalidad*

5.5 Fiabilidad

Hasta dónde se puede esperar que un programa lleve a cabo la función solicitada con la exactitud requerida, cuando se usa bajo condiciones y periodos de tiempo determinados. *Apéndice #9 Evaluación de fiabilidad*

5.6 Eficiencia

La cantidad de recursos informáticos y de códigos necesarios para que un programa realice su función. En otras palabras, se refiere a la cantidad de recursos necesarios para que el producto de software proporcione un desempeño apropiado. *Apéndice #10 Evaluación de eficiencia*

6. Evaluación de etapas, procesos, actividades e hitos

Las etapas, procesos, actividades e hitos deben de ser aprobados una vez terminados, esto permitirá dar por concluido de manera oficial por el probador de software que lo realizado cumple en su totalidad con los requerimientos establecidos.

El contar con un documento que avale la finalización de las actividades ayudará y facilitará al administrador del proyecto el registro de avances. Enseguida se muestra el caso de uso y un ejemplo del formato a utilizar. *Apéndice #10 Evaluación de etapas, procesos, actividades e hitos*

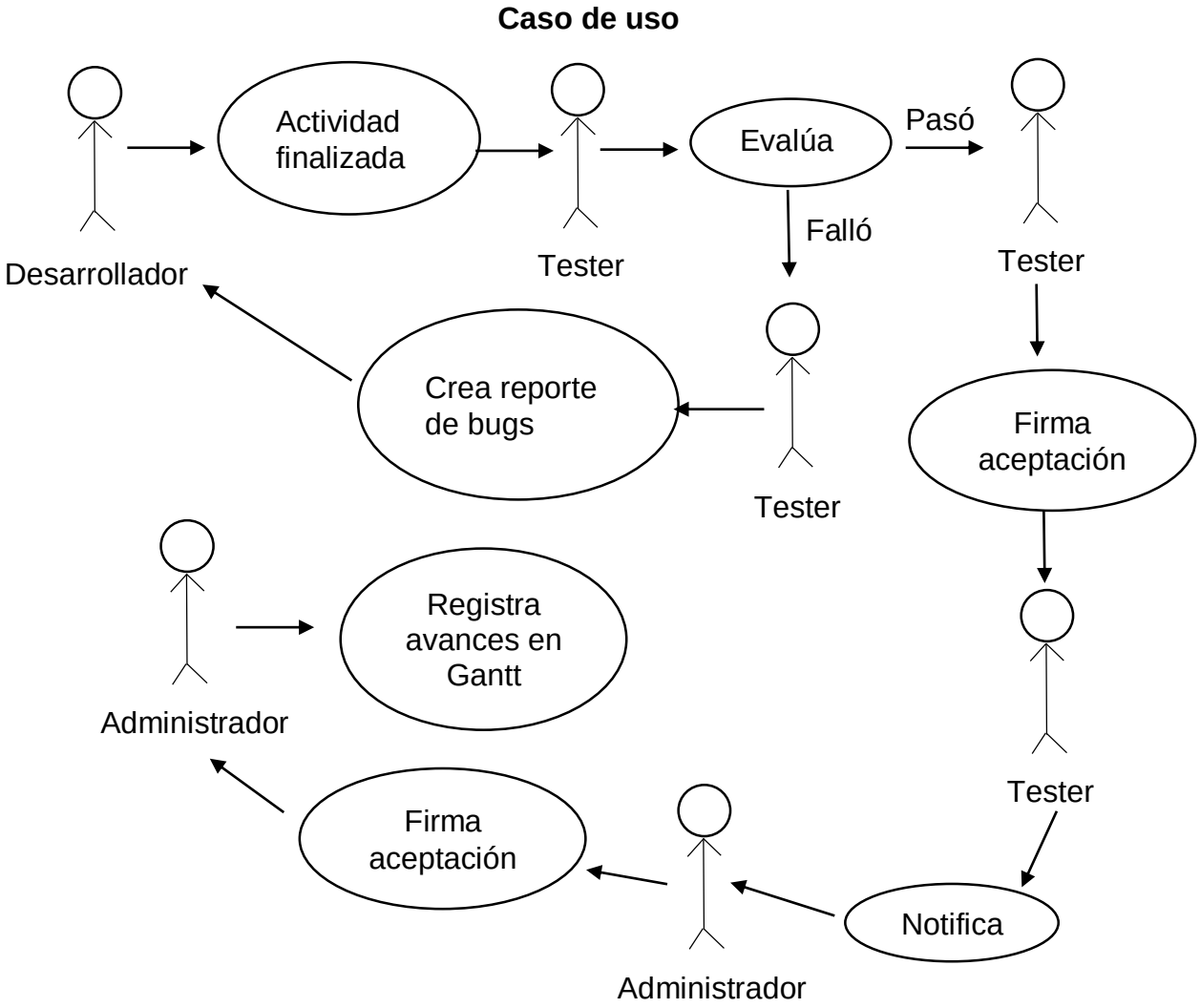

Ilustración 3 Caso de uso para la evaluación de actividades

Ejemplo de la evaluación de la etapa de análisis y diseño

Evaluación de etapas, procesos, actividades e hitos				
Proyecto:	**Sistema para el control de asignaturas**			
Probador de software:	**Juan Carlos López Bustamante**			
Etapa/proceso/actividad/hito	**Fecha estimada**	**Fecha real**	**Firma responsable**	**Firma tester**
1. Análisis y diseño	09/08/2019			
1.1 Análisis de requerimientos	10/07/2019			
1.1.1 Entrevistas	04/07/2019			
1.1.2 Observación	10/07/2019			
1.2 Diagramación	19/07/2019			
1.2.1 Diagrama de clases	12/07/2019			
1.2.2 Diagrama de secuencia	16/07/2019			
1.2.3 Diagrama de casos de uso	19/07/2019			
1.3 Documentación	25/07/2019			
1.3.1 Documento SRS	25/07/2019			
1.4 Diseño de pantallas	09/08/2019			
1.4.1 Ventana login	30/07/2019			
1.4.2 Ventana especialidades	02/08/2019			
1.4.3 Ventana dashboard	07/08/2019			
1.4.4 Ventana instructores	09/08/2019			

_____

Víctor Manuel Hernández Diaz

7. Perfil de un probador de software

Un probador de software revisa un producto de software con la finalidad de recolectar información relacionada con su calidad y el valor que da a quienes lo utilizan.

Este rol toma el desafío de detectar la mayor cantidad de bugs o fallas, esto para evitar que el sistema salga a producción con estos errores. Un probador de software está presente en todas las etapas del proceso de desarrollo, buscando asegurar la máxima calidad del producto final.

Las habilidades con la que debe de contar un probador de software son las siguientes:

- Capacidad de entender y simular el comportamiento del sistema bajo pruebas del usuario final.
- Facilidad de comunicación oral y escrita para interactuar con todos los involucrados en el desarrollo.
- Creatividad para generar ideas e imaginar los problemas que podrían generarse durante el uso.
- Pensamiento crítico para evaluar las ideas, hacer deducciones y vincular lo observado con los criterios de calidad de la empresa.
- Poner en práctica las ideas y adecuar las técnicas y el esfuerzo al alcance del proyecto.
- Aptitudes para el trabajo en equipo, de manera de poder interactuar con los desarrolladores y otros probadores de software, y lograr el máximo beneficio en esta interacción.

APÉNDICES

Apéndice # 1 Evaluación de requerimientos

Revisión de Especificación de Requerimientos de Software			
Fecha:	**Aprobado por:**		
Producto revisado: Software Requirements Specitication (SRS)			
Participantes:			
Nombre	**Rol**	**Iniciales**	**Firma**

Puntos a evaluar	
Criterios	**Comentarios**
Los requerimientos están escritos a un nivel consistente y apropiado de detalles.	
Los requerimientos son lo suficientemente claros.	
Los requerimientos usan la misma terminología.	
Las referencias (interconexiones) entre requerimientos son apropiadas.	
¿Están incluidas todas las funciones que el cliente requiere?	
Cada requerimiento es único y libre de duplicidad.	
Cada requerimiento está correctamente identificado (tiene un identificador).	
Cada requerimiento está dentro del alcance del proyecto.	
Cada requerimiento se puede probar.	
Cada requerimiento establece datos de entrada y salida.	

Cada requerimiento describe sus restricciones.		
¿Los requerimientos de seguridad están correctamente identificados?		
¿Se tienen identificados las dependencias de otros sistemas?		
¿Hay errores ortográficos o gramaticales?		
Resumen de la revisión		
Problema/Sugerencia/Defecto	**Página/Párrafo**	**Prioridad**
Seguimiento		
Acción	**Responsable**	**Fecha entrega**

Apéndice # 2 Formato para revisiones técnicas formales

Revisión técnica formal (RTF)			
Proyecto:			
Historial de revisiones:			
Fecha	**Versión**	**Descripción**	**Autor**
Producto revisado			
Nombre y versión del producto			
Participantes de la revisión			
Nombre	**Rol**	**Iniciales**	**Firma**
Técnicas utilizadas (documentos o formatos)			

Objetivos de la RTF	
Problemas detectados	
Problema	**Sugerencia de corrección**

Evaluación	
Estado actual del producto	

Acciones a tomar	
Acción	**Responsable**

Fecha próxima revisión	

Apéndice #3 Formato para casos de prueba

Casos de pruebas							
Proyecto:							
Autor:		**Fecha:**					
Área funcional:							
Id	**Caso de prueba**	**Descripción**	**Funcionalidad**	**Requerimientos del ambiente de prueba**	**Datos de entrada**	**Datos esperados**	**Datos obtenidos**

Apéndice # 4 Reporte de bugs

Reporte de bugs								
Proyecto:								
Autor:				**Fecha:**				
Área funcional:								
Id	**Título**	**Categoría**	**Descripción**	**Id del caso de prueba**	**Estado**	**Pasos para reproducirlo**	**Resultado real**	**Comentarios**

Apéndice # 5 Evaluación de mantenibilidad

Evaluación de mantenibilidad		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
La modificación de un componente genera un impacto mínimo en el resto del sistema.		
Se puede probar con facilidad.		
Se puede modificar sin provocar defectos o degradar desempeño.		
Se puede reutilizar el código o componente.		
El código es entendible y se identifican las partes a modificar.		
Las variables son nombradas apropiadamente.		
El código está documentado.		

Apéndice # 6 Evaluación de usabilidad

Evaluación de usabilidad		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
Se utiliza un lenguaje sencillo para el usuario (no hay términos técnicos).		
El usuario sabe lo que está pasando en todo momento.		
Existe una interfaz uniforme (comportamiento, diseño).		
Los componentes similares están agrupados.		
Existe una jerarquía en el contenido mostrado.		
La opción de ayuda está disponible en el lugar y momento que se necesita.		
Las opciones seleccionadas por defecto son las indicadas.		
Los mensajes de errores son precisos, dan posibles soluciones.		
Existen mensajes informativos para notificar al usuario de las acciones realizadas.		

Apéndice # 7 Evaluación de portabilidad

Evaluación de portabilidad		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
Se adapta de forma efectiva a diferentes entornos de hardware o software.		
Se puede instalar de forma exitosa en distintos entornos.		
Se puede ejecutar en distintas plataformas.		
La versión puede ser cambiada sin complicaciones.		

Apéndice #8 Evaluación de funcionalidad

Evaluación de funcionalidad		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
El producto cumple con los requerimientos establecidos.		
Los datos de salida son los correctos.		
Los botones cumplen con la tarea indicada.		
La transferencia de datos con sistemas externos ocurre de manera segura y con buen rendimiento.		
Solo los roles autorizados pueden acceder a el producto.		
Los campos de captura tienen la expresión regular correcta.		
Existe un manejo de excepciones para datos introducidos incorrectamente.		

La longitud permitida en el campo de entrada es menor o igual al tamaño del campo de la base de datos.		
Se valida el tipo de dato ingreso (numérico, alfanumérico, solo letras).		
Existe una validación para todos los campos obligatorios.		
No existe código expuesto en el front-end (código cliente).		
Los parámetros enviados a través de la url están encriptados.		
Se cuenta con el protocolo HTTPS para mejorar la seguridad en la transferencia de datos.		
La posición del control (etiqueta, imagen, caja de texto) es la correcta.		
La fuente de datos de un grid se mantiene actualizada después de realizar una modificación de los datos.		
Los datos ingresados no se pierden durante la interacción con la interfaz.		

Apéndice #9 Evaluación de fiabilidad

Evaluación de fiabilidad		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
Al ejecutar una tarea bajo las mismas condiciones se obtienen los mismos datos de salida.		
El sistema mantiene un buen nivel de desempeño bajo las distintas funciones realizadas.		
Tiene la capacidad de evitar fallas ocasionadas por bugs.		
Tiene la capacidad de recuperar desempeño y datos después de un fallo.		
Se encuentra apto para desempeñar una función cuando es solicitado (disponibilidad).		

Apéndice #10 Evaluación de eficiencia

Evaluación de eficiencia		
Proyecto:		
Probador de software:		**Fecha:**
Producto revisado:		
Puntos a evaluar		
Criterios	**Pasó/Falló/NA**	**Comentarios**
Proporciona un desempeño apropiado con relación a la cantidad de recursos utilizados.		
El desempeño se considera que es el esperado de acuerdo a lo pactado.		
Uso apropiado de memoria RAM, disco duro y procesador.		

Apéndice #10 Evaluación de etapas, procesos, actividades e hitos

Evaluación de etapas, procesos, actividades e hitos				
Proyecto:				
Probador de software:				
Etapa/proceso/actividad/hito	**Fecha estimada**	**Fecha real**	**Firma responsable**	**Firma tester**

_____

Nombre y firma administrador

Referencias

Tema	Subtemas y fuente
Concepto de calidad	*Libro:* -Ingeniería del software Un enfoque práctico Séptima Edición (páginas 338-341).
Evaluación de requerimientos	http://www.cs.rug.nl/search/uploads/Teaching/RE2009Fall/project/NASA-Requirements%20Peer%20Review%20Checklist.doc https://fac.ksu.edu.sa/sites/default/files/requirements_review_checklist.doc http://www.pmoinformatica.com/2016/08/tecnicas-analisis-requerimientos.html
Técnicas de revisión	• Revisión técnica formal • Revisión técnica informal Libro: -Ingeniería del software Un enfoque práctico Séptima Edición (páginas 361-365).
Estrategias de prueba para software	• Pruebas de unidad • Pruebas de integración Libro: -Ingeniería del software Un enfoque práctico Séptima Edición (páginas 389-394). • Casos de pruebas https://medium.com/grupo-carricay/qu%C3%A9-son-los-casos-de-pruebas-4893799b5b84 • Categorías de los bugs • ¿Cómo debe de ser un reporte de bugs? https://www.softwaretestingclass.com/wp-content/uploads/2016/06/Beginner-Guide-To-Software-Testing.pdf

Características a evaluar para mejorar la calidad de software	• Mantenibilidad • Usabilidad • Portabilidad • Funcionalidad • Fiabilidad • Eficiencia http://www.sc.ehu.es/jiwdocoj/mmis/externas.htm
Perfil de un probador de software	https://www.ces.com.uy/index.php/ique-es-el-testing/perfil-del-tester-