UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERIA

MAESTRIA EN GERENCIA DE SISTEMAS DE INFORMACIÓN

Plan de Administración de Cambios a los Sistemas de Información de SoftCase

PROYECTO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN GERENCIA DE SISTEMAS DE INFORMACIÓN

Autor: Ing. Ana Patricia Ortíz Coronel

CI: 01034435722

Director: Ing. Christian Hernández Barquero, MSC

Pasaporte: 0108870083

DICIEMBRE, 2016

CUENCA - ECUADOR

RESUMEN

Las empresas desarrolladoras de Software con el tiempo han podido evidenciar que los ajustes realizados sobre sus productos son el punto fundamental de su éxito en el mercado, puesto que dichos ajustes traducen en personalización y mientras un software presente un mayor ajuste a las necesidades de la empresa contratante, es mayormente cotizado.

Debido a esto, en la actualidad existe una tendencia del software a crecer tanto en complejidad como en el tamaño de sus aplicativos, lo cual exige una mayor productividad y excelente calidad en menor tiempo.

Softcase carece de procesos específicos para administrar adecuadamente los cambios que sufren sus productos, por lo que se hace necesario establecer un proceso controlado y ordenado mediante el desarrollo de un Plan de Administración de Cambios. Este plan le permitirá optimizar todos sus recursos y proyectarse hacia un mayor número de clientes, sin descuidar la satisfacción de sus clientes actuales.

Con eso se busca asegurar para la empresa, la optimización tanto de recursos como de costos, proveer de información para determinar si se han alcanzado los objetivos de mejora establecidos dentro de un proyecto, evitar la duplicidad de trabajo y mantener una correcta organización de cada producto, ya sea objeto de base de datos o aplicativo.

Una vez que el Plan de Administración de Cambios sea implementado, es recomendable su automatización mediante alguna herramienta para este propósito, por lo que se realizará una recomendación en base a un breve análisis de tres herramientas mayormente aplicadas en el medio; sin embargo, quedará a consideración de Softcase su manejo.

Palabras Clave: Plan de Administración de Cambios, Administración de Cambios, Gestión de Configuración, Cambios en Software.

ABSTRACT

Software developers over time have been able to prove that the adjustments made to their products are the fundamental point of their success in the market. For this reason, the adjustments have become more personalized and if the Software presents better adjustments according to the needs of the company, it's in greater demand.

Due to this, there is now a tendency for software to grow both in the complexity and size of its applications, which demands greater productivity and excellent quality in less time.

Softcase lacks specific processes to adequately manage the changes suffered by its products, for that reason it became necessary to establish a controlled and orderly process, through the development of a Change Management Plan. This plan allows the company to optimize all of its resources and thus project to a greater number of customers without neglecting the satisfaction of its current clients.

The objective is to ensure that the company optimizes both its resources and its costs, provides information to determine if the improvement objectives established within a project have been achieved, to avoid duplication of work and to maintain a correct organization of each product, either in the database or application.

Once the Change Management Plan is implemented, it is advisable to automate it through any existing tool in the market. So a recommendation will be made based on a brief analysis of three tools mostly applied in the middle; however, it will be for Softcase's consideration.

Keywords: Change Management Plan, Change Management, Configuration Management, Software Changes.

INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

Ana Patricia Ortíz Coronel, autora de la tesis "Plan de Administración de Cambios a los Sistemas de Información de Softcase", reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Magister en Gerencia de Sistemas de Información. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autora.

Cuenca, 20 de diciembre del 2016

Ana Patricia Ortíz Coronel

C.I: 0104435722

Ana Patricia Ortíz Coronel, autora de la tesis "Plan de Administración de Cambios a los Sistemas de Información de Softcase", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autora.

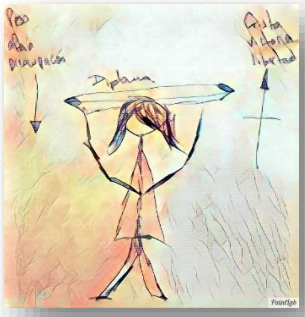
Cuenca, 20 de diciembre del 2016

Ana Patricia Ortíz Coronel

C.I: 0104435722

Dedicatoria

Este proyecto va dedicado a Dios, por ser el pilar donde se fundamenta mi vida; porque en Él me muevo, vivo y soy;
Y por cumplir una promesa más en mi vida.



A mi esposo e hijos, por ser la fortaleza donde se complementa mi vida, mi amor eterno, mi inspiración diaria y mi apoyo incondicional.
A mis amigos, por ser esos hermanos que se escogen y están para animarme siempre.

Ana Patricia.

Agradecimientos

Gracias a la Universidad de Cuenca por impartir valiosos conocimientos que suman al desarrollo personal y profesional de sus alumnos; a Softcase y mis compañeros de trabajo, por permitirme desarrollar y aplicar mi proyecto en aras de aportar a las labores diarias que realizamos para sacar adelante nuestra empresa.

Un agradecimiento especial al Ing. Christian Hernández Barquero por su aporte, colaboración y compromiso con este proyecto sin importar tiempo y distancias.

Gracias a toda mi familia por apoyarme en tiempos de estudio, colaborando y aportando cada uno con su granito de arena en distintos aspectos de nuestro día a día.

Y finalmente gracias Dios, por lo que has Sido, Eres y Serás.

PLAN DE ADMINISTRACIÓN DE CAMBIOS A LOS SISTEMAS DE INFORMACIÓN DE SOFTCASE

Objetivo General

Diseñar *un Plan de Administración de Cambios* para atender las solicitudes de cambio a los productos de Softcase.

Objetivos Específicos

1. Definir los procedimientos relacionados con la administración de cambios que seguirá el personal de SoftCase para atender las solicitudes de cambio de los clientes.
2. Estandarizar uno de los artefactos principales (el Plan de Administración de Cambios) que debe estar presente en cualquier metodología madura de desarrollo de software.
3. Recomendar una herramienta de administración de cambios que permita automatizar los procedimientos de SoftCase para atender de forma estándar los cambios solicitados a sus productos.
4. Dar visibilidad a todo el personal de SoftCase por medio de una documentación clara y detallada, de la definición de los procesos de atención de cambios que deberán seguir para homologar la forma en que trabajan para atender los cambios a los productos en los que están asignados.

CAPITULO 1
INTRODUCCION

1. INTRODUCCION

Este capítulo presenta el objetivo del Plan de Administración de Cambios, su alcance e importancia y además indica las fuentes consultadas para su desarrollo.

1.1 Propósito

El objetivo principal de este proyecto es diseñar un Plan de Administración de Cambios para atender las solicitudes de cambio a los productos de Softcase; que sea sencillo, eficaz y que se adapte a la realidad actual de la empresa, de modo que lo estandaricen dentro de sus políticas para así fortalecer y complementar la metodología de desarrollo que se ha venido aplicando durante todos estos años, asegurando aún más la calidad del producto final que sin duda es el resultante de la calidad del proceso de desarrollo.

1.2 Alcance

Este proyecto se centra en la definición de un Plan de Administración de Cambios para atender los cambios a los sistemas de información que administra y produce Softcase.

Este plan, describirá todas las actividades de configuración y administración de control de cambios que deben realizarse durante el curso del ciclo de vida del proyecto, detallando las responsabilidades asignadas y los recursos requeridos.

Además se analizarán brevemente algunas herramientas que podrían ayudar a su automatización para finalmente recomendar una de ellas, en caso de que Softcase decida institucionalizar la implementación y el uso de los entregables producidos por esta tesis.

1.3 Justificación

Una de las mejores prácticas que ha definido, documentado y madurado la industria del software en los últimos años ha sido todo lo relativo a cómo administrar la configuración y los cambios a los sistemas de información.

La academia, organizaciones y compañías privadas se han visto en la necesidad de invertir recursos para madurar un conjunto de procesos y herramientas que agreguen la robustez requerida para administrar de forma controlada y efectiva los cambios que surgen a los sistemas de información.

Su aplicación estructurada y sobre todo automatizada no es muy común, en especial en Cuenca, por lo que resulta provechoso para Softcase contar con un plan que defina los procesos y herramientas para administrar los cambios a sus sistemas de información de acuerdo con los parámetros que dictan las buenas prácticas de la industria en este campo.

Este proyecto ha nacido como respuesta a una necesidad latente de administrar el constante crecimiento del software, de la complejidad y las relaciones internas que abogan por la presencia de métodos y procedimientos de trabajo formales que permitan comunicar, guiar y armonizar el trabajo que realizan las personas.

Softcase consciente de la importancia de contar con un Plan de Administración de Cambios, se ha comprometido en invertir el tiempo y los recursos que le permitan obtener en el corto y mediano plazo, los siguientes beneficios:

- Formalización y mejoramiento del proceso que sigue su equipo de trabajo para atender todas las solicitudes de cambio a sus productos, de modo que, se base en buenas prácticas que la industria ha sugerido para esta disciplina de la ingeniería de software.
- Estandarización del proceso de administración y control de los cambios. De este modo las órdenes de trabajo (defectos y mejoras a los productos) tendrán un seguimiento adecuado que permita conocer su responsable, esfuerzo, costo y estado en el que se encuentra.

- Desarrollo de guías estandarizadas, documentadas y automatizadas que definitivamente facilitarán la administración de los cambios a los artefactos, dando visibilidad a la organización de la forma en que se incorporan los cambios al software.
- Mejorará la comunicación de todos los miembros del equipo de trabajo, ya que, estarán claramente definidos los canales, responsabilidades, notificaciones y artefactos resultantes del cambio de forma automatizada.
- Proveerá a todos los miembros del equipo, herramientas para la asignación, seguimiento, resolución y puesta en marcha de los cambios que les soliciten, dejando en todo momento rastro del paso a paso que se siguió para atender la solicitud de cambio.

- Permitirá generar métricas que ayuden a mejorar el proceso día con día.

1.4 Definiciones y acrónimos

Back-out. Dar marcha atrás, reversar.

Backlog. Inventario de funcionalidades, mejoras, tecnologías y corrección de errores, que se van incorporando a través de las iteraciones de desarrollo.

CAB. *Change Advisory Board*, Consejo Asesor del Cambio.

CADM. Son las siglas en inglés de CASE *Application Development Method*, que en español significa, Método de Desarrollo de Aplicaciones CASE, es una metodología que se utiliza como herramienta para apoyar el desarrollo de *software*.

CAL. Son las siglas en inglés de *Client Access License*, que en español significa Licencia de Acceso a Cliente, es una licencia que algunos fabricantes de software dan para que un programa sea accedido desde máquinas cliente.

CASE. Son las siglas en inglés de *Computer Aided Software Engineering*, que en español significa, Ingeniería de *Software* Asistida por Computadora, comprende un amplio rango de herramientas que se utilizan para ayudar a las actividades del desarrollo de software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas.

CCB. *Configuration Control Board*, Consejo de Control de la Configuración.

CIs. *Configuration Item*, Elementos de Configuración.

Cluster. Un clúster es un grupo de tablas almacenadas físicamente en conjunto como una sola tabla, que comparten una columna en común. Si a menudo se necesita recuperar datos de dos o más tablas basado en un valor de la columna que tienen en común, entonces es más eficiente organizarlas como un *clúster*, ya que la información podrá ser recuperada en una menor cantidad de operaciones de lectura realizadas sobre el disco.

CMDB. *Configuration Management Database*, Base de Datos de la Gestión de Configuraciones.

CMM. *Capability Maturity Model Integration*, Integración de modelos de madurez de capacidades.

Disparador. En inglés *Trigger,* es un procedimiento que se ejecuta en forma inmediata cuando ocurre un evento especial. Estos eventos sólo pueden ser la inserción, actualización o eliminación de datos de una tabla.

EC. *Emergency Committee*, Comité de Emergencia.

ERP. Son las siglas en inglés de *Enterprise Resource Planning*, que en español significa, Planificación de Recursos Empresariales, es un conjunto de aplicaciones que se engloban en el grupo de Software ERP, que trabajan en forma conjunta, permitiendo una interconexión entre cada uno de los programas pertenecientes al sistema, para obtener un mejor rendimiento empresarial y mayores réditos.

FSC. *Future Schedule of Change*, Calendario de Cambios.

IEEE. *Institute of Electrical and Electronics Engineers*, Instituto de Ingeniería Eléctrica y Electrónica.

Índice. Es una estructura creada para ayudar a recuperar datos de una manera más rápida y eficiente, es creado sobre una o varias columnas de una misma tabla. De esta manera, cuando se solicita recuperar datos de ella mediante alguna condición de búsqueda, ésta se puede acelerar si se dispone de algún índice sobre las columnas-objetivo.

Instancia. Es un conjunto de procesos del servidor Oracle que tiene su propia área global de memoria y una base de datos asociada a ellos.

ITIL. *Information Technology Infrastructure Library*, Biblioteca de Infraestructura de Tecnologías de Información.

Metodología. Es un conjunto de etapas y actividades que conforman el proceso de desarrollo del *software*, en algunos casos también se les puede denominar paradigmas de software que se aplican de acuerdo a la situación y características del sistema o software a desarrollar.

Métrica. Se define como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

ODC. *Orthogonal Defect Classification*, Clasificación Ortogonal de Defectos.

OCG. *Office of Gobernment Commerce*, Oficina de Comercio del Gobierno Británico.

PIR. *Post – Implementation Revision*, Revisión Post – Implementación.

Repositorio. Es un almacén de datos que consta de tablas y vistas que se utilizan como interfaz con los datos y el código procedimental que las maneja. Almacena los detalles del sistema que se está desarrollando, manteniendo *metadatos*, o datos sobre otros datos, como definiciones de objetos o elementos que se están utilizando para crear el diseño del sistema.

RFC's. *Request For Change*, Requerimientos de Cambio.

SCI. *Software Configuration Item*, Elemento de Configuración de Software.

Script. Es un guion o conjunto de instrucciones que se almacena en un archivo de texto plano, ejecutados por un intérprete de línea de comandos, que permiten la automatización de tareas.

SEI. *Software Engineering Institute*, Instituto de Ingeniería de Software.

SLAs. *Service Level Agreement*, Acuerdos de Nivel de Servicio.

Sprint. Subconjunto de requerimientos que serán ejecutados en un período entre 1 y 4 semanas de trabajo.

SQL. Acrónimo de *Structured Query Language* (Lenguaje de Consulta Estructurado). Es un lenguaje especializado de programación que permite realizar consultas a bases de datos.

Storyboad. Registros o bitácoras realizadas por un usuario, sobre un requerimiento que está siendo probado.

SWEBOK. *Software Engineering Body of Knowledge*, Cuerpo de conocimientos de la Ingeniería de Software.

Tabla. Es la unidad lógica básica de almacenamiento. Contiene filas y columnas (como una matriz) y se identifica por un nombre. Las columnas también tienen un nombre y deben especificar un tipo de datos.

TI. Son las siglas en ingles de *Information Technology*, en español Tecnologías de la Información, es un amplio concepto que abarca todo lo relacionado a la conversión, almacenamiento, protección, procesamiento y transmisión de la información. El concepto se emplea para englobar cualquier tecnología que permite administrar y comunicar información.

Vista. Una vista implementa una selección de varias columnas de una o diferentes tablas, no almacena datos; sólo los presenta en forma dinámica. Se utilizan para simplificar la visión del usuario sobre un conjunto de tablas, haciendo transparente para él la forma de obtención de los datos.

1.5 Referencias

- [1] Michael R. Lyu, *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill Company, 1996.
- [2] Brian A. White (Intro by Geoffrey M. Clemm), *Software Configuration Management Strategies and Rational ClearCase: A Practical Introduction* (TheADDP9 Object Technology Series). Addison Weszey Professional, 2000.
- [3] Gu Yang, *Adopting ODC to improve software quality: A case study.* Rational Edge, 2006.
- [4] Beth Chrissis Mary, Konrad Mike, Shrum Sandy, CMMI® Guía para la integración de procesos y la mejora de productos. Pearson Educación, 2009.
- [5] Abran Alain, W. Moore James, Guide to the Software Engineering Body of Knowledge. Computer Society, 2005.
- [6] Lijnse, Peter, Above and Beyond ITIL Change Management. Hewlett-Packard Company, 2002.
- [7] Czap Kevin, Integrated Change and Release Management. IBM, 2008.
- [8] Roger Pressman, Ingeniería de Software un enfoque práctico 7ma. edición, McGraw-Hill Company, 2010.
- [9] IBM, Rational Unified Process, IBM, 2006.
- [10] IBM Research, Orthogonal Defect Classification, http://researcher.watson.ibm.com/researcher/view_group.php?id=480, Acceso 20 de septiembre del 2016
- [11] Carnegie Mellon University, Orthogonal defect classification applied to a multidisciplinary design, http://researcher.watson.ibm.com/researcher/view_group.php?id=480, 1996, Acceso 20 de septiembre del 2016.

- [12] Modelo de Capacidad y Madurez, https://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez, Acceso 28 de septiembre del 2016
- [13] Project Tools – Herramientas para la gestión de proyectos, https://projectools.wordpress.com/modelos-de-madurez-en-gestion-de-proyectos/, Acceso 28 de septiembre del 2016
- [14] ITIL Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/registro_del_cambio.php, Acceso 29 de septiembre del 2016.
- [15] Pierre Bourque, Richard E. Fairley, SWEBOK V3.0 Guide to the Software Engineering Body of Knowledge, 2014
- [16] Software Engineering Institute, CMMI for Development, Version 1.3, Noviembre 2010
- [17] IBM Rational ClearQuest, http://www-03.ibm.com/software/products/en/clearquest, Acceso 24 de octubre del 2016
- [18] Rational ClearQuest, https://en.wikipedia.org/wiki/Rational_ClearQuest, Acceso 23 de octubre del 2016
- [19] Atlassian JIRA Software, https://www.atlassian.com/software/jira, Acceso 25 de octubre del 2016
- [20] JIRA, https://es.wikipedia.org/wiki/JIRA, Acceso 25 de octubre del 2016
- [21] Microsoft Developer, Steven Lange, Requirements Management in TFS, https://blogs.msdn.microsoft.com/slange/2007/11/06/requirements-management-in-tfs-part-1-of-4-overview/, Acceso 25 de octubre del 2016
- [22] Microsoft, TeamFoundation, https://msdn.microsoft.com/en-us/library/ms181232(v=vs.90).aspx, Acceso 26 de octubre del 2016
- [23] Microsoft, TeamFoundation Overview, https://msdn.microsoft.com/en-us/library/ms242904(v=vs.90).aspx, Acceso 26 de octubre del 2016

1.6 Visión General

El presente proyecto se centra en el desarrollo de un Plan de Administración de Cambios ajustado a las necesidades de la empresa Softcase.

Busca analizar las ventajas que se obtienen de llevar un adecuado control de cambios dentro de una empresa desarrolladora de Software, con el objetivo de que Softcase implemente el plan desarrollado y así, éste le permita optimizar costos y agilizar sus procesos y tiempos de respuesta.

Para lograr este objetivo, se evaluará lo que dictan las buenas prácticas y modelos de madurez como CMMI, ITIL y SEWBOK, los mismos que complementados con herramientas existentes en el mercado, aseguran una sencilla y óptima aplicación del Plan de Administración de Cambios.

Finalmente se analizarán brevemente tres de las mejores herramientas del mercado para manejo de control de cambios, IBM RationalClearQuest, JIRA de la compañía Atlassian y Microsoft TeamFoundation Server (TFS); con el objetivo de realizar una recomendación a Softcase sobre cuál de ellas le resultaría más conveniente.

CAPITULO 2
CONTEXTO DE LA ORGANIZACIÓN

2. CONTEXTO DE LA ORGANIZACIÓN

Este capítulo da una breve reseña de la empresa Softcase y los métodos de desarrollo que utiliza para sus sistemas.

2.1 Contexto Histórico

Softcase, Software Consultoría Asesoramiento y Soluciones Empresariales, es una Compañía Limitada fundada en 1999, dedicada al desarrollo e implementación de software principalmente hospitalario; nace como resultado de la unificación de esfuerzos de un grupo de ingenieros quienes formaron parte de uno de los más ambiciosos proyectos informáticos dentro de la Ciudad. El proyecto inicial fue lanzado en 1997 como una iniciativa Universidad-Empresa que intentaba automatizar toda la labor que se realiza dentro del Instituto del Cáncer Solca - Cuenca.

Una vez terminado exitosamente este proyecto, se funda la compañía y desde entonces ha venido desarrollando e implementando sistemas con resultados altamente positivos.

Organigrama

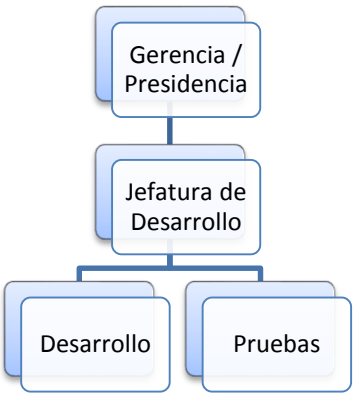


Ilustración 1: Organigrama de la Empresa Softcase

2.2 Objetivos

Desarrollar programas computacionales, con la finalidad y capacidad de resolver situaciones reales a las empresas para su desarrollo y evolucionamiento tecnológico, para atender sus necesidades y hacerlas mejores en su campo laboral, financiero, administrativo y económico.

2.3 Productos y servicios

SoftCase Es una empresa pionera en la utilización de herramientas CASE de ORACLE para el diseño y desarrollo de aplicaciones.

Entre sus principales productos están:

- Software de Control Médico
- Software Administrativo Financiero
- Software de Administración Farmacéutica

Los mismos que integrados conforman un ERP, lo suficientemente robusto, estable y testeado para seguir ganando mercado, dentro y fuera de la ciudad.

Modelo y metodología de desarrollo

Los procesos de diseño y construcción de los sistemas de información de Softcase, están apoyados en herramientas de software usadas como ayuda durante todo el proceso. Tanto analistas, diseñadores así como desarrolladores, utilizan programas para capturar información sobre los requisitos del negocio, a partir de esto se crean diseños para las estructuras de datos que satisfagan dichos requisitos y finalmente se genera el código de los programas frontales y del servidor.

Este conjunto de herramientas se encuentra centralizado en Designer, el CASE de ORACLE, que contempla el método de desarrollo de aplicaciones CADM, en el cual se tiene un proceso que abarca el camino de un requisito de negocio desde su fuente original producida normalmente durante el análisis, hasta su eventual implementación en el sistema. La clave de su éxito reside en asegurarse que cada fase ha sido completada correctamente antes de pasar a la siguiente.

Designer cuenta con un repositorio de tablas y vistas que se utilizan como interfaz con los datos y el código procedimental que las maneja, almacena los detalles del sistema que se está desarrollando además de mantener metadatos, como definiciones de objetos o elementos que se están utilizando para crear el diseño del sistema. Estos objetos pueden ser entidades y sus atributos o tablas y sus columnas, los cuales tienen aspectos individuales llamados propiedades que los definen.

Designer es de gran impacto sobre el método de desarrollo, ya que influye directamente en todas las fases del proyecto y permite utilizar las herramientas adecuadas para cada etapa; además ayuda a realizar análisis y diseños más óptimos proporcionando un repositorio unificado para almacenar la información de estas etapas. Aunque CADM no pone énfasis en los prototipos, Designer permite generar pequeños prototipos como parte de las fases de análisis y diseño con el objetivo de mostrar una prueba de un concepto determinado.

Fases

El método de desarrollo utilizado consta de las siguientes fases:

1. Estrategia
2. Análisis
3. Diseño
4. Construcción
5. Pruebas

6. Implementación
7. Mantenimiento

En cuanto a la documentación, no es simplemente un paso separado al final, sino implica un proceso continuo a lo largo de todo el proceso de desarrollo del sistema, de esta forma no se convierte en una etapa más que alargue el ciclo de vida e implique trabajo y tiempo adicionales a la entrega del proyecto.

1. Fase de Estrategia

Esta fase está totalmente centrada en el negocio, su objetivo principal es obtener una buena comprensión de las metas de cada área, objetivos, procesos, dirección y necesidades del negocio para estructurar y documentar la visión del proyecto. Perfila el alcance del proyecto y define un acuerdo con el cliente.

Entregables

- Términos de Referencia
- Diagrama Entidad / Relación de estrategia
- Diagrama de Flujos de Procesos de estrategia

Herramientas de Apoyo

Modelador de Procesos

El propósito de esta herramienta es crear una representación visual de las actividades del negocio, muestra procesos, flujos de información y las unidades organizativas a las que corresponden o se ven afectadas por ellos. Permite descomponer un proceso de alto nivel en otros de menor nivel, realizando un diagrama de subprocesos.

Además de brindar apoyo a esta fase, también proporciona soporte a las fases de Análisis y Diseño.

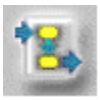


Diagrama de Flujo de Procesos

Muestra los procesos del negocio y sus interacciones, no se detalla mayormente, a menos que el cliente lo requiera. Básicamente es usado para demostrar al cliente que el equipo de desarrollo comprende plenamente los procesos del negocio dentro del alcance del proyecto.

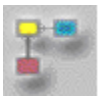


Diagrama Entidad / Relación

El propósito de este diagrama es identificar los datos primarios o entidades principales del área de negocio. Su objetivo es definir las necesidades del usuario al más alto nivel.

En esta fase, este diagrama plasma una idea poco elaborada de las entidades básicas, pudiendo o no contener atributos

Mientras que el Modelador de Procesos abarca la cara funcional del área del negocio, el Diagrama Entidad / Relación se concentra en la parte de los datos, los mismos que son esencialmente lógicos, es decir no necesariamente existirán tablas físicas reales que sustenten cada una de las entidades.

2. Fase de Análisis

Esta fase se centra en obtener todas las especificaciones del usuario para el proyecto y detallar completamente los procesos del negocio que estarán implicados en el diseño del sistema.

Inicialmente se realiza el levantamiento de información mediante entrevistas, observación de procesos actuales, revisión de informes y documentos necesarios para el giro del negocio, además de revisiones del sistema actual si existiera.

En esta fase, se especifican por completo el Diagrama Entidad / Relación, el Diagrama de Flujos de Procesos y se suma el Diagrama de Jerarquía de Funciones para organizar los requisitos del usuario en áreas funcionales, en éste se debe realizar las posibles mejoras en las funciones que principalmente vayan a estar referenciadas en módulos de la aplicación. Cada requisito debe estar plasmado en una o más funciones o entidades, los flujos de procesos deben estar bien detallados y mostrar tanto los del negocio como los propuestos.

Posteriormente se genera el documento que contiene la especificación de requisitos, que será la base principal de la Fase de Diseño. Un documento con requisitos inadecuados o sin concluir puede provocar la insatisfacción del cliente o un fallo en el sistema.

Finalmente se revisa nuevamente el Análisis con ayuda de la jerarquía de funciones, ésta debe mostrar los procesos del negocio y asegurarse que todos los requisitos del sistema están planificados para su implementación. Si todas las principales funciones del negocio están representadas, todos los requisitos están haciendo referencia al menos a una función y cada función tiene al menos un requisito asociado, la evaluación y verificación del análisis se da por concluida.

Entregables

- Diagrama Entidad / Relación de análisis
- Diagrama de Flujos de Procesos de análisis
- Diagrama de Jerarquía de Funciones
- Especificación de Requisitos

Herramientas de Apoyo

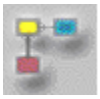


Diagrama Entidad / Relación

Descrito anteriormente.

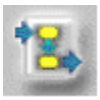


Diagrama de Flujo de Procesos

Descrito anteriormente.

Diagrama de Jerarquía de Funciones

Esta herramienta se emplea para refinar la jerarquía de procesos, asociar usos de entidades a las funciones, añadir usos de atributos y permite modelar procesos durante la fase de análisis. Se emplea generalmente en conjunto con el Diagrama Entidad / Relación, de hecho contiene sus mismas funciones pero presenta un aspecto de cuadro organizativo.

Su objetivo principal es mostrar las funciones del sistema de un modo jerárquico o multinivel, ofreciendo una visión del negocio basada en la lógica sin profundizar en grupos de usuarios, departamentos o almacenes de datos, permitiendo manipular la jerarquía misma.

Inicialmente se encuentra creado un diagrama de jerarquía de funciones para cada proceso padre, el cual es generado automáticamente a partir de los datos almacenados en el repositorio sobre las funciones hijas del proceso raíz y es sobre esta base que se realizan las modificaciones necesarias.

RON
Navegador de Objetos del Repositorio

Esta herramienta permite el acceso y manipulación de las propiedades de cualquier elemento del repositorio, se usa para visualizar las definiciones de los elementos creados durante la fase de estrategia y permite cambiar o añadir cualquier información que se pueda mejorar o haya sido omitida respectivamente.

Informes del Repositorio

Esta herramienta permite visualizar mediante reportes, las definiciones de los elementos en formatos particulares, con esta utilidad pueden ejecutarse más de 150 informes predefinidos.

Para esta etapa los utilizados son los reportes de las definiciones de entidades y flujos de datos. En caso de no encontrar un informe que satisfaga las necesidades, se puede tomar uno de ellos como base para modificarlo o en su defecto crearlo desde el inicio y atarlo a esta herramienta.

3. Fase de Diseño

En esta fase se genera un diseño conceptual y físico de las aplicaciones, el cual debe incluir una lista de módulos que se van a desarrollar, referenciando los requisitos del sistema hacia dichos módulos.

Antes de comenzar a construir el sistema se deben afinar todos los detalles puesto que la visión del desarrollo usando CASE es muy irreal de primera mano, por ello se dedica tanto tiempo a la preparación de los módulos para su generación y su depuración posterior, que es muy importante para tener una visión clara de lo que se quiere obtener.

Esta fase se divide en dos partes:

1. Diseño de la base de datos

   Se diseñan tablas y columnas con sus especificaciones detalladas de atributos, claves primarias, dependencias, dominios, restricciones, disparadores (triggers), columnas redundantes, distribución de espacio y la desnormalización necesaria para afinar el rendimiento de la base.

   Además se diseñan tablas y vistas de acumulación y resumen, así como también tablas de descripciones de códigos.

2. Diseño de la aplicación

   Se diseñan las aplicaciones del software, incluyendo las estructuras que soporten los módulos, programas, procedimientos, navegación, controles internos y seguridad, además de definir los usos de cada columna y elaborar un completo listado de funcionalidades de cada módulo y sus componentes (Libro de diseño); Con estas especificaciones se valida que el diseño de la base de datos se haya completado correctamente durante la creación del diseño conceptual.

   Ya que en esta etapa se referencian los módulos hacia la base de datos, se permite validar la misma, además de identificar columnas perdidas, redundantes o innecesarias.

   El diseño físico de las pantallas se realiza mediante un esquema manual, para posteriormente ser generado dentro del repositorio en la fase de Construcción, deben ser construidas de acuerdo a lo que puede ser generado desde Designer pues a pesar de ser una herramienta muy poderosa, no es muy flexible al momento de generar diseños complejos.

Finalmente el Diseño es sometido a una evaluación y aceptación conjunta del jefe de proyectos con el cliente.

Entregables

1. Diseño de la base de datos
2. Diseño conceptual de aplicaciones
3. Diseño físico de aplicaciones
4. Libro de Diseño
5. Documento de aceptación del Diseño

Herramientas de Apoyo

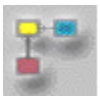


Diagrama Entidad / Relación

Descrito anteriormente.

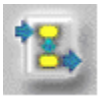


Diagrama de Flujo de Procesos

Descrito anteriormente.

RON
Navegador de Objetos del Repositorio

Descrito anteriormente.

Informes del Repositorio

Descrito anteriormente.

4. Fase de Construcción

Esta fase al igual que la anterior comprende dos procesos principales los cuales deberían fluir sin ningún inconveniente, si todas las fases anteriores han sido cuidadosamente desarrolladas.

Estos procesos son:

1. Construcción de la base de datos

   Esta construcción supone su generación directa mediante Designer, incluyendo las estructuras de datos, dominios especificados, disparadores y todos sus componentes.

   Este proceso es automático y muy sencillo, el único trabajo específico que hay que realizar en esta etapa es definir la cantidad de espacio reservado para cada tabla y su localización física; en este momento se define las instancias[1] de Oracle a utilizar que generalmente son dos, una para Desarrollo / Pruebas y otra para Producción.

2. Construcción de aplicaciones

   La construcción de los módulos requiere de gran trabajo dado que se deben generar a partir de un repositorio nada gráfico para su creación, por ende el tiempo que tome su desarrollo dependerá mucho de la experiencia del desarrollador y el uso correcto de los generadores.

   Estos generadores son herramientas muy potentes que usadas en orden y de manera correcta, generan automáticamente las aplicaciones listas para funcionar de una manera perfecta, pero de no ser así pueden incrementar e inclusive doblar el tiempo de desarrollo.

[1] Una instancia es un conjunto de procesos del servidor Oracle que tiene su propia área global de memoria y una base de datos asociada a ellos.

Generador para el Servidor. Como primer paso se deben crear los objetos de la base de datos para generar con éxito el código para un módulo. Este generador es una utilidad del repositorio que produce archivos de texto de SQL*Plus que pueden ejecutarse para crear los objetos de la base de datos (Scripts).

Generador para Forms. Crea archivos de aplicación de pantallas y menús, basándose en las definiciones del repositorio; Y bloques e ítems de formularios, en base a los usos de las tablas y columnas asociadas a cada módulo.

Finalmente se realizarán los ajustes necesarios y pruebas a nivel de unidad para asegurar la corrección funcional y técnica, antes de ser pasado al departamento de Pruebas.

Estas evaluaciones de unidad son corridas conforme se construye la aplicación puesto que todavía se tiene en mente la idea inicial de todo lo que se requiere para satisfacer las especificaciones del Libro de diseño.

Entregables

1. Base de Datos
2. Aplicaciones

Herramientas de Apoyo

Editor de Diseño

Esta herramienta se divide en cuatro secciones que realizan diversas tareas como construcción y generación de la base de datos, manipulación de datos sobre la jerarquía de objetos, pueden incluirse columnas o restricciones para claves primarias, creación de índices, clusters, vistas, entre otras.

Es similar al Diagrama Entidad / Relación, la diferencia es que mientras que este muestra el modelo conceptual, el Editor de Diseño mediante su componente *Diagramador del Modelo del Servidor* muestra el diseño lógico que es lo que al final se implementará.

Cuenta con un generador de código que recorre el repositorio y genera las sentencias SQL que crean los objetos diseñados en una base de datos relacional, en caso de que los objetos ya estén creados la herramienta utiliza sentencias *ALTER TABLE* para continuar con su generación.

Dentro de sus componentes también está el Diagramador de módulos, el cual en su modo de *Datos,* muestra el uso de los datos dentro de un módulo, permite visualizar las tablas o vistas que estarán dentro de un formulario o informe y las relaciones entre ellos. En el modo de *Vista,* el diagramador permite recoger el formato gráfico de los módulos, indica cómo se organizará visualmente a nivel de ventanas, vistas fijas, botones, navegación entre otros; pero todo de forma estructural mas no gráfica.

5. Fase de Pruebas

Esta fase es una de las más importantes, ya en esta instancia cuando surgen errores, no es necesario revisar los diseños lógico o físico de la base de datos, sino netamente las aplicaciones, en cuanto a si satisfacen los requisitos del usuario y cumplen con la funcionalidad especificada en el Libro de diseño.

En primera instancia se realizan pruebas de unidad, cada aplicación debe ser meticulosamente evaluada, usando datos de prueba, inspecciones de código y despejando dudas conjuntamente con el desarrollador del aplicativo.

Posteriormente se efectúan pruebas globales y de correlación tanto a nivel de módulos como de base de datos, que de ser satisfactorias se emitirá un documento que respalde su aprobación.

Finalmente se realizan pruebas de aceptación de los usuarios, en donde se proporcionan las aplicaciones para que trabajen con ellas y se realicen transacciones reales, de modo que se asegure su correcto funcionamiento y la aceptación de los resultados obtenidos plasmados en un acta de entrega recepción.

Entregables

1. Documento de aprobación del departamento de Pruebas
2. Acta de entrega y recepción satisfactoria del Usuario

Herramientas de Apoyo

Sistema Desarrollado

6. Fase de Implementación

Antes de entrar a esta Fase, previamente se imparte la capacitación requerida en cuanto a funcionamiento y soporte de la aplicación, al personal que se encargará del Centro de Cómputo de la empresa contratante, de esta forma estarán listos para encargarse del soporte a usuario y los refuerzos requeridos en cuanto a capacitación del personal.

El punto inicial es la capacitación al personal que manejará los distintos módulos que comprende el sistema, se entregan manuales de funcionamiento del sistema y se programan inducciones por tipos de usuarios.

Para la puesta en marcha, en caso de no existir previamente un sistema, se instala el aplicativo y se lo pone en marcha iniciando por las configuraciones iniciales que sean requeridas para comenzar con su funcionamiento global.

Si existe un sistema anterior, se realiza un análisis de viabilidad de migración de datos, de ser posible se detiene dicho aplicativo en cierto punto y se realiza una migración de los datos requeridos para posteriormente poner en funcionamiento el sistema desarrollado y continuar trabajando dentro del mismo, abandonando por completo el sistema anterior; En caso de no ser posible dicha migración, se pone en funcionamiento el aplicativo y es decisión del cliente las acciones a tomar en cuanto a unificación de información.

Entregables

1. Manuales de Usuario
2. Actas firmadas de capacitaciones impartidas

Herramientas de Apoyo

Sistema Desarrollado

Manuales de usuario

7. Fase de Mantenimiento

Aun cuando el sistema está "finalizado" y puesto en marcha, está sujeto a cambios continuos pues como en todo sistema nuevo surgen inconvenientes que generan diversas necesidades como incremento de privilegios de usuarios, solicitudes de nuevos informes, solicitudes de cambios en funcionamiento, entre otros.

Cabe recalcar que muchos de estos cambios no solo afectan a las aplicaciones sino también a la base de datos por lo que es muy importante mantenerlos documentados y realizar una actualización a los manuales entregados y por ende puede requerir de capacitación a los usuarios.

Debido a que esta etapa se desarrolla netamente en el entorno empresarial y que los cambios requeridos son prioritarios para continuar con el funcionamiento del sistema, los desarrolladores entran a trabajar directamente con los usuarios y es aquí en donde el ciclo normal del desarrollo se rompe, generando un grave inconveniente dentro del control de cambios, puesto que son aplicados directamente en los módulos del sistema y no se registran al tiempo ni la documentación, ni las actualizaciones encadenadas, requeridas para la integridad del sistema de producción. Esta tarea pasa a segundo plano hasta cuando el sistema quede netamente depurado, implicando trabajo y tiempo considerable, sin contar con que pueden existir omisiones o confusiones al momento de registrar las actualizaciones generadas.

2.4 Organización

Actualmente la empresa tiene una forma particular y empírica de administrar los cambios a los sistemas de información que ha desarrollado, esta le ha permitido controlar la evolución que éstos van sufriendo durante el tiempo, producto de la corrección de sus fallas y por supuesto de las nuevas funcionalidades que se añaden para satisfacer las necesidades de sus clientes.

Sin embargo, con cada nuevo sistema que se desarrolla, se incrementa la complejidad de la administración de los cambios, debido a:

- El aumento y especialización del personal involucrado en el desarrollo y mantenimiento de los mismos.
- Las relaciones entre los sistemas existentes, en términos de las funcionalidades, interfaces y componentes base que se comparten.
- El constante desarrollo de nuevos componentes de software que se convierten en la base de su arquitectura de software.
- El crecimiento en el número de clientes que han adquirido los productos de SoftCase

Estos aspectos, producen que la forma empírica con la que se han administrado los cambios se haya vuelto insuficiente, complicada y con fallas en el control y seguimiento de los cambios que se atienden para los diferentes productos desarrollados por SoftCase.

Sin duda alguna, es una necesidad para la empresa, contar un proceso de administración de cambios formal, estandarizado y apoyado sobre herramientas automatizadas, pues así, todos sus miembros podrán homologar la forma en que deben atenderse todas las solicitudes de cambio, garantizando siempre el adecuado control y seguimiento que se requiere para brindar alta calidad en los productos que utilizan sus clientes.

Por estas razones, proveer a SoftCase de un *Plan de Administración de Cambios* que describa detalladamente el proceso que se seguirá para administrar los cambios de forma estandarizada y automatizada, permitirá aumentar su nivel de madurez y alinearla a las mejores prácticas que hoy dicta la industria del software.

En cuanto al alcance de este proyecto, la tesis se centra en la definición de un *Plan de Administración de Cambios* que defina el proceso que debe seguirse para atender los cambios a los sistemas de información que administra y produce SoftCase., además de recomendar una herramienta que podría ayudar a la automatización del mismo.

CAPITULO 3
MARCO TEORICO

3. MARCO TEORICO

Este capítulo aporta con conceptos básicos a cerca de las bases sobre las cuales se desarrolla el Plan de Administración de Cambios.

3.1 [2]Concepto de administración de cambios

El cambio es un componente básico del software computacional, combina procedimientos humanos y herramientas para llevarlo a cabo, cuando se solicita un cambio, este debe ser analizado y registrado antes de ser implementado, debe ser reportado a quienes deben estar en conocimiento del mismo y por supuesto debe mantener un control, todo esto con el objetivo de no causar confusión entre los ingenieros de software involucrados en un proyecto, ya que la incorporación de esos cambios al software de forma descontrolada, frecuentemente provoca caos.

El proceso de solicitud de cambios nos permite dar seguimiento a las solicitudes de cambio y efectuar mediciones acerca de la actividad de cada una de ellas. Una vez que una solicitud es recibida, se procede a registrarla y realizar una evaluación técnica (análisis de impacto) se lleva a cabo para determinar qué modificaciones se requerirían y si la solicitud de cambio debe ser aprobada. Tener un buen entendimiento de las relaciones entre los elementos de software es importante para llevar a cabo el análisis de impacto. Finalmente, una autoridad establecida evaluará todos los aspectos del cambio propuesto y, aprobará, modificará, rechazará o pospondrá dicho cambio. La autoridad para aprobar o rechazar los cambios generalmente se le conoce como comité de control de cambios.

[2] [1] [2] [3] [5] [7] [8]

Según Bersoff, la primera ley de la ingeniería de sistemas afirma: “No importa dónde se encuentre en el ciclo de vida del sistema, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida”[3]; De ahí que la Administración de Cambios es una actividad que se realiza durante todo el proceso de desarrollo, cuya meta principal es maximizar la productividad, minimizando los errores.

El control de cambios debe guardar un equilibrio, de lo contrario ya sea con un control exagerado o un control mínimo, se podrían generar problemas en un software mediano; para un proyecto de software mayor, el cambio descontrolado conduce rápidamente al caos. Para tales proyectos, el control del cambio combina procedimientos humanos y herramientas.

Las fuentes más frecuentes del cambio son:

a. Nuevas condiciones en el negocio
b. Nuevas necesidades del cliente
c. Reorganización o crecimiento del negocio
d. Restricciones presupuestales
e. Optimización del sistema

Las actividades en el proceso de la Administración de Cambios, se desarrollan para:

a. Identificar el Cambio
b. Controlar el Cambio
c. Garantizar que el cambio se realizara de manera adecuada
d. Reportar los cambios a todos los interesados

[3] Bersoff, 1982

Todos los involucrados en el desarrollo de un proyecto deberían en alguna medida participar en la Administración de Cambios, aunque hay empresas con más recursos que tienen personas especializadas para este fin. El hecho que todos los integrantes estén en conocimiento de los cambios, ayuda a eliminar la posibilidad de confusiones que pueden resultar de alto costo para el proyecto y asegurar que no existan inconsistencias en el sistema desarrollado, generadas por:

- Actualizaciones simultáneas. Cuando varios integrantes del equipo trabajan sobre un mismo elemento a la vez.
- Problemas en la notificación de cambios. Cuando un problema fue resuelto para algún elemento que es compartido por varios desarrolladores y alguno de ellos no fue notificado de dicho cambio.
- Múltiples versiones.

Cuando cualquier producto puede explicarse, seguirse y controlarse, cuando los cambios pueden seguirse y analizarse y cuando todos los interesados están enterados de un cambio, significa que el mismo fue realizado exitosamente.

Los elementos del control de cambios son:

- Solicitudes de cambio (RFC's)
- Autoridad de control de cambios
- Orden de Ingeniería de cambios
- Procesos de alta y baja de objetos
- Nivel de cambio, informal (antes de la línea base), a nivel de proyecto (cambio local) o formal (cuando el software está instalado)

El proceso formal del Control de Cambios es el siguiente:

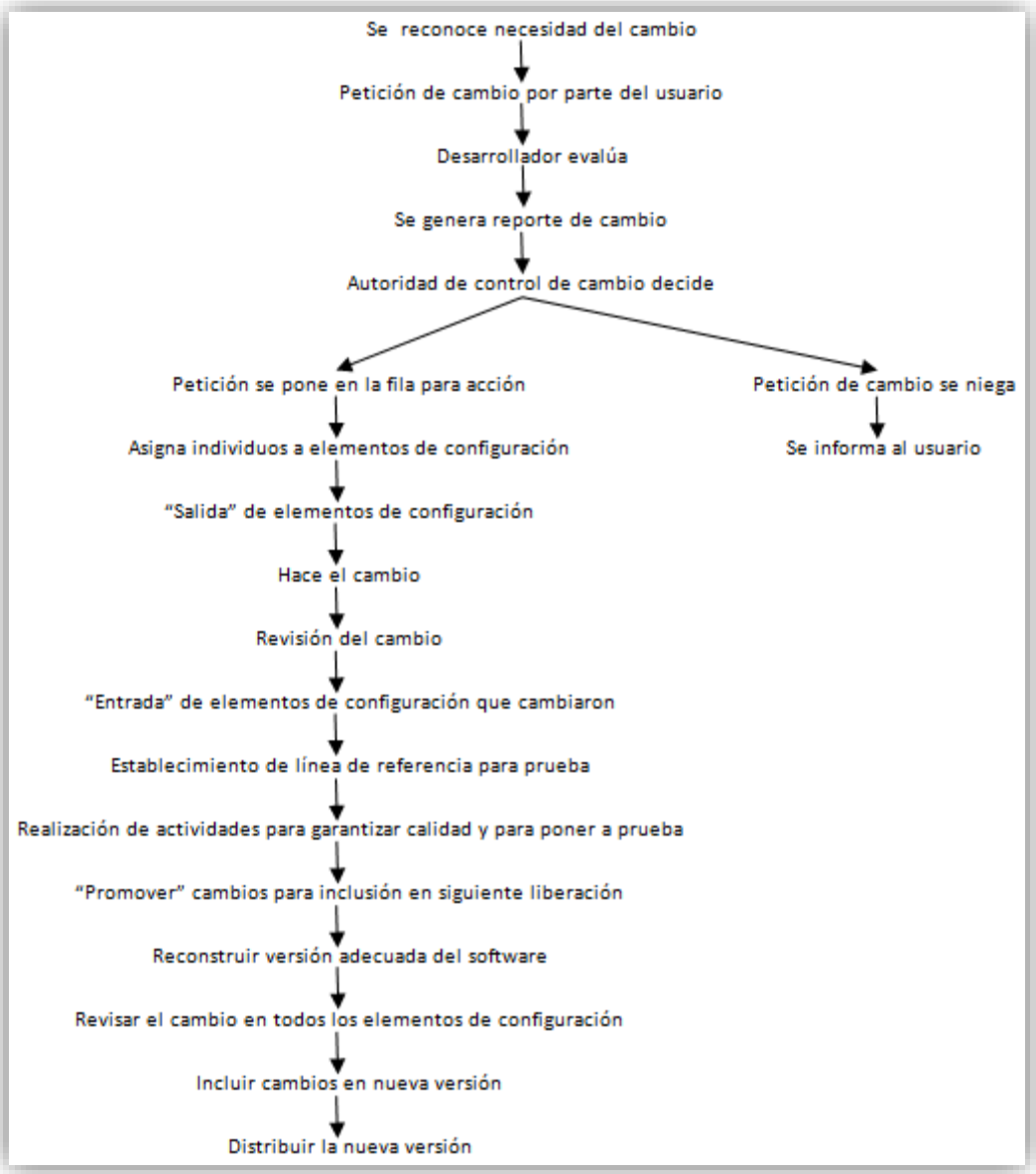


Ilustración 2: Proceso formal de la Administración de Cambios (Adaptado de [8])

3.2 [4]¿Por qué administrar los cambios?

El desarrollo de software sin control puede llegar a generar diversos inconvenientes tales como retrasos en la entrega de productos, generación de productos de mala calidad, entre otros; por lo que el mantener una adecuada Administración de Cambios es una sólida práctica de la ingeniería de software, cuyo principal objetivo es la evaluación y planificación del proceso de cambio para asegurar que, si éste se lleva a cabo, se haga de la forma más eficiente, siguiendo los procedimientos establecidos y asegurando en todo momento la calidad y continuidad de los sistemas.

Debe ser considerado como una actividad paralela al desarrollo del proyecto que responde a eventos que surgen del mismo, sea por requerimientos propios del usuario o por mejoras o correcciones detectadas por el mismo equipo del proyecto, puede ser aplicada indistintamente a proyectos en marcha o proyectos ya implementados, y es necesario resaltar su importancia y no relegarla como una actividad posterior al desarrollo, sino reconocerla como una actividad que debe estar definida, presente y es crítica desde el inicio del proyecto, puesto que si el cambio no se controla en un proyecto de software, puede ser una causa de caos, más aún cuando hay una gran cantidad de personas involucradas en el desarrollo del mismo.

El Control de Cambios es esencial al momento de tener control sobre todos los elementos generados por los integrantes del equipo de proyecto. Este control ayuda a eliminar la posibilidad de confusiones que pueden resultar de alto costo para el proyecto y asegurar que no existan inconsistencias en el sistema desarrollado, generadas por:

- Actualizaciones simultáneas: Cuando varios integrantes del equipo trabajan sobre un mismo elemento al mismo tiempo.

[4] [1] [2] [4] [8] [9]

- Problemas en la notificación de cambios: Cuando un problema fue resuelto para algún elemento que es compartido por varios desarrolladores y alguno de ellos no fue notificado de dicho cambio.

Algunos de los beneficios que se obtienen de la realización de un buen Control de Cambios son:

- Brindar apoyo a los métodos de desarrollo de software
- Mantener la integridad del producto
- Asegurar que los elementos bajo configuración, son completos y correctos
- Proveer un ambiente estable y controlado de trabajo
- Restringir y controlar los cambios que se realizan
- Proveer mecanismos de rastreo de porqué, cuando y quién realizó un cambio

Cuando se desarrolla y/o mantiene un proyecto es vital reducir la probabilidad de que un cambio afecte la disponibilidad de los servicios de TI mediante el análisis de impacto y la evaluación del backlog de cambios en forma conjunta.

De igual manera con un control adecuado de los cambios, se pueden utilizar de forma más conveniente los recursos disponibles que generalmente son austeros, evaluando su tratamiento en función del costo / beneficio para el negocio; para ello es importante también el establecimiento de políticas, almacenamiento de históricos de cada cambio e identificar la dependencia entre los mismos, de manera que podamos evaluar la incidencia de modificación sobre un mismo aplicativo o procedimiento de base de datos.

3.3 [5]Clasificación ortogonal de defectos

Antecedentes

La Clasificación Ortogonal de Defectos identificada como ODC por sus siglas en inglés, surgió hace un poco más de veinte años en el centro de investigación Thomas J. Watson de IBM. Inicialmente estaba enfocado a proveer retroalimentación a los desarrolladores y a las personas involucradas en la verificación y validación del software, a partir de los datos relevantes sobre defectos encontrados durante el proceso de producción.

La primera aparición del esquema completo fue en el año 1992 y fue refinándose hasta llegar a su última versión que es la 5.2 publicada en septiembre del 2013.

Inicialmente ODC se enfocó en establecer las bases para proveer de un análisis y retroalimentación de defectos orientado a problemas de calidad en el diseño del software y código en un entorno de lenguaje de procedimiento; y posteriormente fue complementado para cubrir todas las áreas del desarrollo de software incluyendo especificaciones propias de la programación orientada a objetos. Su metodología ha sido aplicada tanto en empresas grandes como pequeñas dentro de diversas aplicaciones.

Definición

ODC es una técnica utilizada para categorizar los defectos encontrados en el proceso de desarrollo de software, que permite medir de manera cuantitativa y con recursos mínimos los aspectos que deben ser tomados en consideración de forma prioritaria durante la etapa de desarrollo, con el fin de tomar medidas correctivas que eviten la aparición de defectos similares en el futuro.

[5] [10] [11]

Su principal aplicación es para la mejora del desarrollo del software, ya que los defectos son clasificados en distintas clases que colectivamente apuntan al área del proceso de desarrollo en específico que requiere de atención.

Esta clasificación debe ser establecida de forma genérica de manera que pueda ser usada en todas las etapas del desarrollo de software y debe cumplir con los siguientes requerimientos:

- Ortogonalidad, es decir sin superposición de las clases, esto es, que un defecto solo puede estar bajo una única clase
- Consistencia a través de las fases
- Uniformidad a través de los productos

ODC puede ser usada para mejorar el círculo de calidad del proceso de prevención de defectos, lo cual se ha demostrado que resulta en un alto grado de ahorro en costos de análisis ya que el proceso de clasificación permite encontrar de forma retrospectiva la causa y efecto en un muy corto tiempo en relación al tiempo que toma a un grupo de analistas efectuar un detallado análisis de la causa raíz del defecto.

Esta técnica ayuda a cerrar la brecha entre métodos cuantitativos y análisis cualitativos. En la mayoría de casos, cuando se analizan los defectos se escribe la información en lenguaje natural, el cual no es cuantitativo. ODC proporciona un sistema para convertir esa información en métricas que proporcionen estadísticas que permitan definir de manera más acertada los puntos que necesitan ser tratados, resultando así ser un instrumento para detallar mejor y recibir la retroalimentación más oportunamente.

Generalmente nos hacemos preguntas como ¿en dónde están los problemas del producto? o ¿desde qué punto nacen los defectos?, pero al ser preguntas tan generales, no se puede obtener una respuesta puntual y efectiva pues dependerá del conocimiento, experiencia o intuición del desarrollador, dichas respuestas serán de utilidad como un análisis macro del sistema pero no pueden ser consideradas como medidas para el análisis de la efectividad del proceso.

El primer paso es crear clases de defectos en función de las especificaciones del proyecto, el objetivo es que dichas clases puedan explicar el progreso del producto a través del proceso y que puedan ser utilizadas a lo largo de ciclo de desarrollo del proyecto.

Declaración de clases o tipos de defectos

Los tipos de defectos deben definir o deben capturar en su nombre, el significado de lo que se arregló, por ejemplo se declaran clases como verificación, declaración de funciones, interfaz o rendimiento.

Una vez definidos los tipos de defecto deben analizarse y para ello se debe mapear el tipo de defecto con el proceso, independientemente del estado en el que se encuentre y del producto que se utiliza, lo importante es encontrar la relación entre los tipos de defecto y la etapa del proceso en la que se podría detectar.

Los pasos para definir una clase son:

1. Definir las métricas necesarias
2. Validar los puntos a corregir tomando en consideración la experiencia en otros proyectos
3. Mejorar el sistema de mediciones cuando se aprenden nuevos métodos a través de los experimentos

Una vez definidas las clases, deben determinarse los atributos o condiciones que se usarán para definir los defectos puntuales, deben ser adecuados para poder categorizar cada uno de ellos de manera correcta y mapearlos a la etapa en la que pudo haberse prevenido el error.

Definición de atributos

Existen dos tipos de atributos que se recomiendan para la definición de errores:

Tipo de defecto: Se declaran los tipos donde un programador pueda categorizar un defecto de manera sencilla y sin confusiones. Es recomendable usar los siguientes tipos de defecto:

Código faltante: Se refiere a que la funcionalidad no fue implementada, por lo que la verificación encontró el defecto.

Código incorrecto: Hace referencia cuando el código fue implementado pero tiene errores que llevaron a encontrar el defecto.

Errores de función: Se refiere a problemas de capacidad, configuración o comunicación entre capas de la arquitectura.

Asignación: Pocas líneas de código, como inicialización de variables.

Interface: Errores con la interfaz final del usuario o interacción con otros componentes.

Validación: Lógica del sistema que falló en validar configuración erróneas.

Sincronización: Manejo de recursos compartidos.

Build, empaquetado y merge: Errores con librerías externas utilizadas, manejo de cambios o control de versiones.

Tipo de disparadores: Un disparador o trigger, es una condición que saca a la superficie un defecto. El concepto no proporciona una perspectiva del proceso de desarrollo, sino del proceso de verificación. Idealmente, la distribución de los disparadores para los defectos de campo debe ser similar al encontrado en las pruebas del sistema. Una discrepancia puede indicar deficiencias en el ambiente de pruebas.

Clasificación de Causas y Efectos

Cuando se detecta un defecto en cualquier etapa del desarrollo del proyecto, es importante recolectar la información no solo de los disparadores sino también de todos aquellos inconvenientes derivados del error que afectan directamente al desempeño del aplicativo.

Causa: Atributo de clasificación ortogonal que describe un defecto para:

- El proceso de desarrollo
- El proceso de verificación

Efecto: Atributos o métricas que están afectando al producto o a los procesos, por ejemplo:

- Áreas de impacto
- Densidad de defectos
- Rehacer arreglos de código

Planteamiento general del método

ODC permite estimar el avance de un proyecto extrayendo información semántica de los defectos. En el contexto de la metodología, un defecto se refiere a un cambio necesario en el software, básicamente categoriza los defectos en clases, y señala que parte del proceso requiere atención. Al corregir un defecto, el programador le asigna un "tipo" (en el sistema de seguimiento de errores), que es trazado a una o varias fases del proceso de desarrollo (diseño, programación y pruebas). Añadir una nueva característica (defecto funcional) es distinto a cambiar un par de líneas de código para corregir el valor de una variable (defecto de asignación). Al ser la clasificación un proceso manual, se busca que el conjunto de tipos de defectos sean ortogonales (es decir que no se solapen entre ellos) para minimizar los errores y evitar confusiones. Además, los tipos de defectos deben ser generales para que sean independientes del tipo de desarrollo, de las fases del mismo e incluso del producto.

Los tipos de defectos se asocian a una o más fases del proceso de desarrollo, como se muestra en la Tabla 1. En este ejemplo los defectos funcionales se asocian con el diseño, y se espera poder detectarlos además en la inspección de alto nivel y las pruebas funcionales. De esta manera, la asociación indica donde se espera un pico en defectos de tipo funcional. Por tanto, la tabla de mapeo describe el perfil de defectos en cada una de las fases.

	Funcionales	Validación	Sincronización	Algoritmo
Diseño	X			
Diseño de bajo nivel			X	
Implementación		X		X
Inspección de alto nivel	X			
Inspección de bajo nivel			X	
Inspección del código		X		X
Pruebas unitarias		X		X
Pruebas funcionales	X			X
Pruebas del sistema			X	

Tabla 1: Asociaciones entre fases y tipos de defecto (Adaptado de [11])

En la Ilustración 2 se muestra gráficamente cómo cambian los perfiles de defectos en un proyecto en las diferentes fases del desarrollo. Este ejemplo en particular usa sólo cuatro tipos de defectos: funcional, de asignación, interfaces y de sincronización, y para cada fase se muestra la distribución normalizada del tipo de defectos. El comportamiento esperado de la gráfica va a depender del proceso de desarrollo. En la mayoría de los procesos de desarrollo se hace un diseño antes de iniciar la implementación y las pruebas, por lo que se espera que los defectos funcionales sean mayores al inicio del proyecto, y disminuyan conforme se avanza. Por otra parte, es razonable esperar que los problemas de sincronización sean encontrados durante la prueba del sistema.

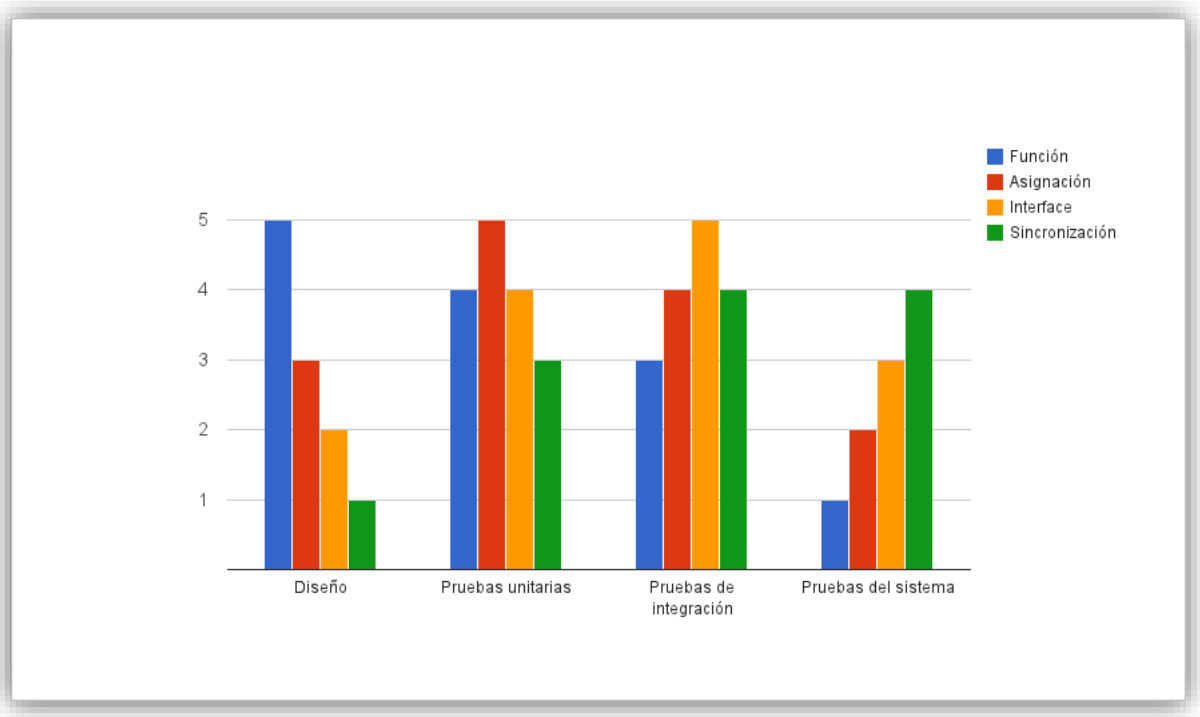


Ilustración 3: Cambio en la distribución de tipos de defectos según las fases (Adaptado de [11])

Así, un cambio en la distribución de los tipos de defectos va a proporcionar una medida del progreso del proyecto. Al mismo tiempo, permite validar si el desarrollo se encuentra "lógicamente" en el mismo lugar en el que está "físicamente". Por ejemplo, si se están ejecutando las pruebas del sistema, pero el perfil de la distribución se parece al perfil de pruebas unitarias, indica que se están ejecutando las pruebas del sistema prematuramente.

Es ilustrativo comparar el tipo de información que se obtiene de la distribución de tipo de defectos contra un indicador como el "total de defectos", como se muestra en la Ilustración 3. La gráfica muestra un incremento constante en los defectos, pero es difícil determinar cuándo se perdió el control del proyecto. Para este mismo proyecto, la Ilustración 3 muestra la proporción de defectos de tipo funcional para periodos de 6 meses. Aquí es fácil ver como el porcentaje de defectos de tipo funcional creció constantemente, cuando es uno de los problemas que debería atacarse en una etapa temprana del desarrollo.

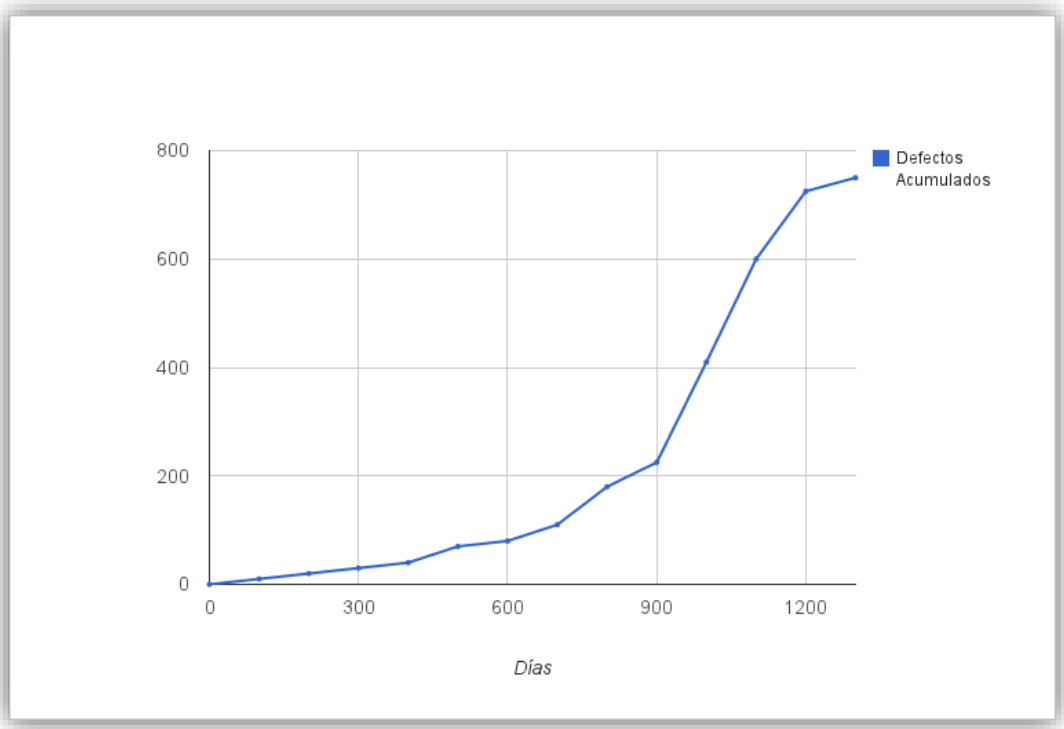


Ilustración 4: Cantidad de defectos contra el tiempo (en días) (Adaptado de [11])

3.4 [6]Buenas prácticas y modelos de madurez y su enfoque en la administración de cambios

Un modelo de madurez es un conjunto de buenas prácticas, criterios de análisis y evaluación, herramientas de medición, entre otras, mediante las cuales se puede determinar las capacidades de un proyecto dentro de una organización, para compararlas con estándares y así identificar las falencias actuales y establecer procesos de mejora continua para buscar la excelencia en la administración de proyectos.

La base de los modelos de madurez es el Modelo de Madurez de Capacidades, conocido como CMM por sus siglas en inglés, el mismo que fue desarrollado por el centro de investigación de la Universidad Carnegie – Mellón, denominado Instituto de Ingeniería de Software (SEI) en 1986, con el objetivo de evaluar los procesos vinculados con el desarrollo de software, proveyendo de un cuestionario a manera de herramienta para identificar las áreas que necesitan cualquier tipo de mejora.

Entre los principales y más relevantes sobre todo en la Administración de Cambios están:

[7]CMMI

La Integración de modelos de madurez de capacidades, conocida como CMMI por sus siglas en inglés, es una guía de ayuda en la mejora continua de procesos, la cual puede ser aplicada y complementada para ajustar a las necesidades de los proyectos, su enfoque permite cambiar de un proceso de caos a un proceso ordenado, estandarizado, medido y optimizado, que establece una base para la mejora continua, permitiendo a la organización adoptar nuevas prácticas sobre los procesos establecidos.

[6] [12] [13]

[7] [16]

Es una evolución de CMM, su primera versión fue publicada en el año 2001 y la última en liberarse fue la versión 1.3 en el año 2010 y al igual que CMM fue generado en el Instituto de Ingeniería de Software (SEI) de la Universidad Carnegie – Mellón.

Es un modelo de madurez de mejora de procesos para el desarrollo de productos y servicios, consiste en las mejores prácticas que tratan las actividades de desarrollo y mantenimiento que cubren todo el ciclo de vida del software.

Administración de Cambios

El control de cambios parte de un requerimiento, producto de una necesidad de mejora, de una necesidad de modificación o de la presencia de un defecto, independientemente de cuál sea el origen, cada solicitud de cambio debe ser parte del proceso de control para mantener la calidad del software, evitar redundancia de trabajo y optimizar costos.

CMMI establece ciertos pasos para una adecuada administración:

1. En primera instancia se deben registrar las solicitudes de cambio (RFC), incluyendo todos los procesos que estén embebidos dentro del mismo ya que sobre la totalidad se evaluarán ciertos aspectos tales como la cantidad de cambios incluidos, complejidad en cuanto a las áreas de proyecto que se ven involucradas, estimaciones de tiempo y costos, entre otras.
2. Se debe analizar el impacto que los cambios provocarán sobre todos los componentes o aplicativos ya existentes del proyecto, su consistencia con los requisitos técnicos, las eventualidades que puedan presentarse, así como las posibles inconsistencias con la solución planificada de antemano; puesto que un cambio además de solucionar algún inconveniente puede ser el causante de muchos otros.

3. Como siguiente paso se debe priorizar y categorizar la solicitud de cambios, si se identifica un cambio de Emergencia se deben remitir a la autoridad de emergencia si es pertinente. Cada cambio deberá ser integrado como parte de una próxima línea base.

4. Posteriormente deben revisarse las RFC's que serán tratadas en la siguiente línea base, llegando a un consenso con las partes involucradas en la solicitud de cambio. Esta revisión debe realizarse con las personas apropiadas quienes darán paso al registro de la disposición efectuada sobre cada solicitud, el motivo que justifica dicha decisión, incluyendo criterios de éxito, si es requerido además podrán efectuar un plan de acción y las necesidades que deben ser o no satisfechas con la implementación del cambio. Una vez solventados estos puntos deberán ser comunicados a las partes interesadas y solicitantes del cambio.

5. Finalmente los cambios deben ser implementados de manera eficiente y oportuna, tomando en consideración las recomendaciones realizadas durante el proceso de análisis. Adicionalmente debe realizarse un seguimiento de las RFC's hasta su cierre con la aprobación necesaria, deben ser procesadas en su totalidad en cuanto sea posible, ya que el hecho de tener RFC's acumuladas sin un proceso completo tiene una directa implicación sobre los costos de la organización sin pasar por alto la confusión que pueden generar entre los desarrolladores.

[8]ITIL

La Biblioteca de Infraestructura de Tecnologías de Información, conocida como ITIL por sus siglas en inglés, es un estándar de calidad, un conjunto de buenas prácticas para la gestión de servicios, desarrollo y operaciones relacionadas a las tecnologías de la Información, provee de procedimientos de gestión detallados para ayudar a las organizaciones a lograr calidad y eficiencia en las operaciones de TI independientemente de su proveedor, ITIL fue diseñado como una guía que engloba toda la infraestructura, desarrollo y operaciones de TI.

Aunque ITIL es de libre utilización, pertenece a la Oficina de Comercio del Gobierno Británico (OCG), se desarrolló a finales del año 1980 y su última versión (V3) se publicó en el año 2011.

ITIL define un cambio como "la adición, modificación o eliminación de un servicio o componente autorizado, planificado o de soporte y su documentación relacionada". Son eventos aprobados por la gerencia, llevados a cabo con el menor riesgo posible para la infraestructura.

[8] [14] [6]

Proceso de la administración de cambios en ITIL

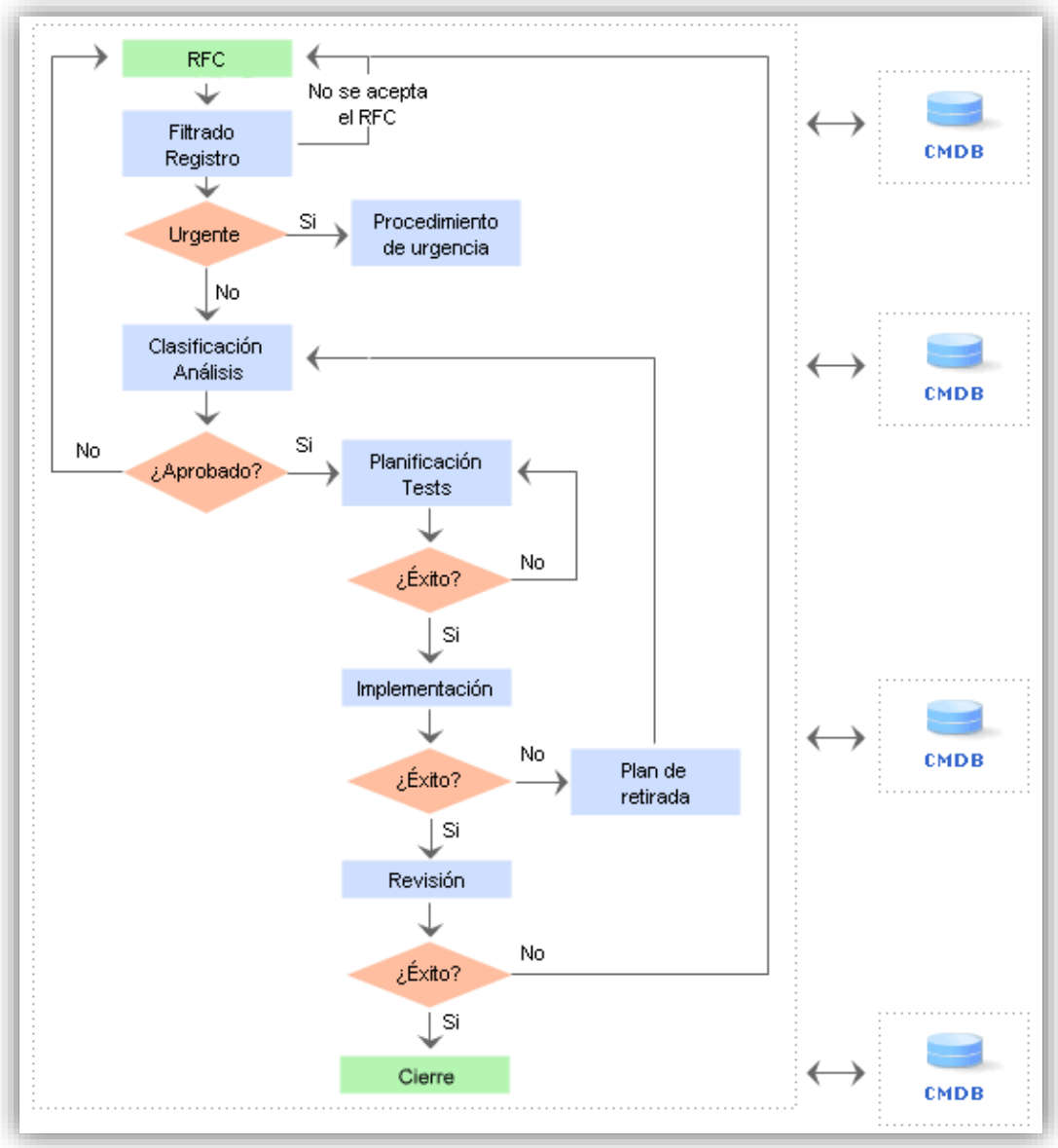


Ilustración 5: Proceso de la Administración de cambios ITIL (Tomado de [12])

Registro. El proceso de cambios parte del registro adecuado de una solicitud de cambio, generalmente para cambios de baja importancia o repetitivos pueden establecerse procedimientos generales que no requieran de una aprobación de la gestión de cambios; pero cuando se tratan de requerimientos más complejos, las solicitudes de cambio son indispensables y pueden ser de distintos tipos:

- Gestión de problemas. Solucionar un error ya conocido y generalmente implica un cambio en la infraestructura de TI.

- Nuevos servicios. Se trata de nuevos desarrollos, los que en su mayoría requieren cambios en la infraestructura de TI y por tanto debe ser coordinado con todas las áreas involucradas de manera que se analice su pertinencia, sus objetivos y que su creación no deteriore la calidad de otros servicios prestados.

- Estrategia empresarial. La dirección de la organización puede solicitar cambios que se ajusten a nuevos objetivos, en cuyo caso siempre implican cambios en hardware, software y/o procedimientos.

- Actualizaciones de software de terceros. Si se da el caso que la organización maneje software de otros proveedores los cuales dejen de brindar soporte en versiones actualmente instaladas o lancen nuevas versiones, será necesario eventualmente una actualización para ajustarse a los nuevos requerimientos.

- Imperativo legal. Cuando se requieren cambios para ajustarse a una nueva medida económica o cambio de legislación (por ejemplo porcentaje del IVA), pueden exigirse cambios en la infraestructura TI.

- Otro. Cuando cualquier empleado, cliente o proveedor puede sugerir mejoras en los servicios que pueden requerir cambios en la infraestructura.

Aceptación. Luego del registro de la solicitud de cambio se debe evaluar su pertinencia, si está justificado el cambio que requieren o si de pronto puede ser mejorada o complementada con una mejor definición, en caso de ser rechazada deberá ser devuelta al departamento desde donde surgió de modo que puedan hacer correcciones al requerimiento o en su defecto sustentar más ampliamente su necesidad.

Clasificación. Después de su aceptación se debe clasificar al requerimiento, otorgándole una prioridad y una categoría dependiendo de la urgencia y del impacto que pueda tener.

La prioridad marca la relevancia y la importancia que tiene el requerimiento sobre los demás que se encuentran pendientes y es un dato relevante para conformar el calendario de cambios.

La categoría marca la dificultad y el impacto del requerimiento de cambio, por lo tanto será determinante al momento de asignar los recursos necesarios, plazos y niveles de autorización para su respectiva implementación, su determinación se basa en el impacto sobre la organización y el trabajo requerido para su implementación. Los niveles básicos de prioridad son:

- Baja. Cuando pueden ser implementados conjuntamente con otros cambios del mismo tipo, o por ejemplo cuando se decidan actualizar ciertos paquetes de software o se compre nuevo hardware, etc.
- Normal: Este tipo de cambios conviene realizarlos en un momento oportuno cuando su riesgo sea minimizado y no interfiera con la implementación de algún otro cambio de prioridad superior.
- Alta: Cuando un cambio debe realizarse sin demora alguna, generalmente están asociados a errores conocidos que deterioran considerablemente la calidad del servicio. El Consejo Asesor del Cambio (CAB) debe evaluar este cambio y ejecutar los procesos necesarios para proveer de una solución.
- Urgente: Este tipo de cambios requieren una solución inmediata, pues generalmente provocan una interrupción o dificultades graves en el servicio. Un cambio de prioridad urgente desencadena un proceso denominado cambio de emergencia.

En general, los cambios menores no requieren de una autorización previa para su implementación, para los demás es necesario un análisis y la debida aprobación del Consejo Asesor del Cambio (CAB).

Aprobación y Planificación

La implementación de cualquier cambio por más pequeño que sea, debe pasar por un proceso de planificación previa, puesto que de no ser así por más sencillo que fuera, podría causar graves problemas, es por ello que es de suma importancia contar siempre con planes de "back out" que permitan recuperar la última configuración estable.

Para la aprobación de cambios el CAB se reúne periódicamente con el objetivo de analizar cada una de las RFC's, su impacto y su probable aceptación, en caso de cambios de alto impacto deberá consultarse con la dirección ya que pueden verse implicados aspectos de carácter estratégicos o de políticas generales de la organización.

Los aspectos evaluados por el CAB son los siguientes:

- ¿Cuáles son los beneficios esperados del cambio propuesto?
- ¿Justifican esos beneficios los costes asociados al proceso de cambio?
- ¿Cuáles son los riesgos asociados?
- ¿Se dispone de los recursos necesarios para llevar a cabo el cambio con garantías de éxito?
- ¿Puede demorarse el cambio?
- ¿Cuál será el impacto general sobre la infraestructura y la calidad de los servicios TI?
- ¿Puede el cambio afectar los niveles establecidos de seguridad TI?

Una vez aprobados los cambios se decidirá si deben ser implementados individualmente o dentro de un grupo de cambios para optimizar el proceso de implementación como uno solo, en cuyo caso se obtienen ciertas ventajas tales como:

- Optimización de recursos necesarios
- Supresión de posibles incompatibilidades entre cambios

- Elaboración de un solo plan de “back out”
- Simplificación del proceso de actualización de la CMDB y la revisión post – implementación

Implementación

Con respecto a la implementación ITIL le asigna a la Gestión de Cambios únicamente a supervisión y coordinación del proceso, ya que la implementación como tal no recae dentro de sus responsabilidades.

Durante el desarrollo del cambio se debe asegurar que:

- Tanto hardware como software, adquirido o desarrollado respectivamente, se ajusten a las especificaciones predeterminadas
- Se cumplan los calendarios establecidos y que la asignación de recursos sea la adecuada
- El entorno de pruebas es realista y simula adecuadamente el entorno de producción
- Los planes de "back-out" permitirán la rápida recuperación de la última configuración estable

De ser posible, debe permitirse el acceso restringido de usuarios al entorno de pruebas para que realicen una valoración preliminar de los nuevos sistemas en lo que respecta a su funcionalidad, usabilidad y accesibilidad.

La opinión de los usuarios durante el proceso de implementación debe ser tomada en cuenta y de ser pertinente, la RFC debe ser revisada en caso de que se encuentren objeciones justificadas al cambio.

Evaluación

Previo al cierre del cambio es de vital importancia realizar una evaluación que permita determinar el impacto de las modificaciones realizadas, tanto en la calidad de servicio como en la productividad de la organización. Para ello se deben considerar los siguientes aspectos:

- ¿Se cumplieron los objetivos previstos?
- En qué medida se apartó el proceso de las previsiones realizadas por la Gestión de Cambios
- ¿Provocó el cambio problemas o interrupciones del servicio imprevistas?
- ¿Cuál ha sido la percepción de los usuarios respecto al cambio?
- ¿Se pusieron en marcha los planes de "back-out" en alguna fase del proceso? ¿Por qué?

Si la evaluación final determina que el proceso y los resultados han sido satisfactorios se procederá al cierre de la RFC y toda la información se incluirá en la Revisión Post – Implementación asociada.

Cambios de Emergencia

Cuando existen situaciones emergentes que afecten de gran manera a los sistemas críticos de la organización o a varios usuarios, se debe actuar inmediatamente siguiendo por supuesto un procedimiento de emergencia debidamente previsto, por ejemplo se pueden definir ciertos protocolos tales como:

- Una reunión urgente del Consejo Asesor del Cambioy/o Comité de Emergencia si esto fuera posible
- Una decisión del Gestor del Cambio si es imposible demorar la resolución del problema o éste sucede durante un fin de semana o periodo vacacional (lo que puede dificultar la reunión del Comité de Emergencia)

Ya que el objetivo primordial es restablecer los servicios, suele suceder que los procesos asociados sigan un orden poco habitual, es decir que la documentación y los registros de cambio sean realizados posteriormente a la implementación de la solución; sin embargo es de vital importancia que el proceso de cierre sea el establecido, con el objetivo de evitar nuevas incidencia y problemas inmediatos.

Control del Proceso

Para finalizar correctamente el proceso se deben elaborar ciertos informes que permitan evaluar mediante información precisa y sencilla, el rendimiento de la Gestión de Cambios, para ello se deben elaborar ciertas métricas tales como:

- RFC's solicitados
- Porcentaje de RFC's aceptados y aprobados
- Número de cambios realizados clasificados por impacto y prioridad y filtrados temporalmente
- Tiempo medio del cambio dependiendo del impacto y la prioridad
- Número de cambios de emergencia realizados
- Porcentaje de cambios exitosos en primera instancia, segunda instancia, etc.
- Número de back-outs con una detallada explicación de los mismos
- Evaluaciones post-implementación
- Porcentajes de cambios cerrados sin incidencias ulteriores
- Incidencias asociadas a cambios realizados
- Número de reuniones del CAB con información estadística asociada: número de asistentes, duración, nº de cambios aprobados por reunión, etc.

[9]SWEBOK

El Cuerpo de conocimientos de la Ingeniería de Software, conocido como SWEBOK por sus siglas en inglés, es una guía que describe el conocimiento existente de la ingeniería de software, su desarrollo inició en 1998 como un proyecto de la IEEE que consideró la necesidad de su publicación para que la ingeniería del software sea reconocida como una disciplina legítima y una profesión. Su última versión es la 3.0 publicada en el año 2014.

Administración de Cambios

Para clarificar el proceso que establece SWEBOK para la administración de cambios, es necesario definir ciertos elementos y conceptos que forman parte de dicho proceso.

Un *Elemento de Configuración de Software* conocido por sus siglas en inglés como SCI, es un objeto sobre el cual recaen actualizaciones o modificaciones, cada SCI debe ser identificado con sus atributos y relaciones ya sea con módulos, listas, entre otros, siendo sumamente necesario documentarlos de manera adecuada.

El *Consejo de Control de la Configuración,* es la autoridad responsable de aceptar, rechazar o recomendar modificaciones a los cambios solicitados.

La *Línea base* de un software, es un conjunto de elementos de configuración (SCI), que se han fijado formalmente en un punto determinado del ciclo de vida del software, representa la configuración actual aprobada, la cual no puede moverse sin un procedimiento de control de cambios formal y plenamente justificado y documentado. El término se usa para referirse a una versión establecida de un elemento de configuración.

[9] [15]

El flujo que plantea SWEBOK para realizar una efectiva administración de cambios es el que se detalla en la ilustración 5, el cual incluye todo el proceso partiendo desde una solicitud de cambio, su análisis, la autoridad requerida para aprobar algunos de ellos y el soporte respectivo para su correcta implementación, además de las desviaciones o cancelaciones de algunos otros.

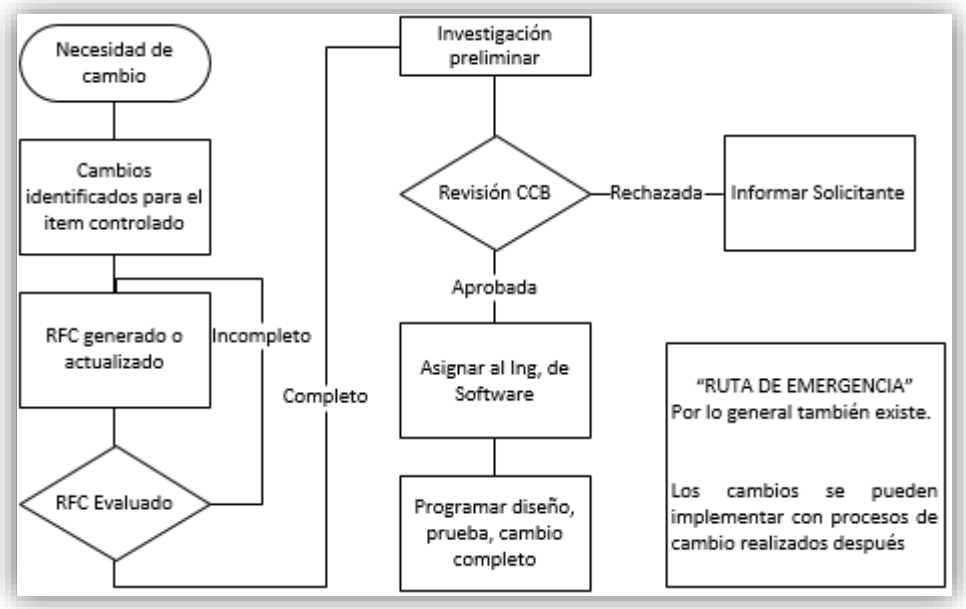


Ilustración 6: Flujo del Proceso de Administración de Cambios SWEBOK (Tomado de [15])

Con respecto a la petición, evaluación y aprobación de cambios, SWEBOK ha determinado procedimientos formales para receptar y registrar las solicitudes de cambio RFC's, evaluando su origen, el impacto tanto económico como funcional con respecto a otros componentes ya instalados y finalmente aceptando, rechazando o recomendando una modificación del requerimiento presentado.

Cuando ingresa una solicitud de cambio, se debe registrar el tipo de cambio, es decir si es un defecto o una mejora, esto sirve para medir la incidencia por tipo de cambios y determinar posibles causas orígenes de los mismos. Una vez que se registra una solicitud, se realiza una evaluación técnica o análisis de impacto para determinar el alcance que deberá tener el cambio y su incidencia tanto en componentes de software como de hardware que se vean involucrados durante el proceso. Finalmente la evaluación profunda y de gestión será realizada por la autoridad establecida, de acuerdo con la línea base y el elemento de la configuración involucrados en el cambio, ésta decidirá si se aprueba, modifica, rechaza o pospone el cambio solicitado.

En lo que concierne específicamente al proceso de petición de cambios, se requiere del uso de herramientas de apoyo, ya sean formularios de papel, un diagrama de procedimientos documentado o herramientas electrónicas para gestionar todo el proceso; desde la generación de un requerimiento, su flujo de proceso, las observaciones realizadas por el CCB y emitiendo informes completos sobre todo el proceso de cambio.

Para la implementación de los cambios que hayan sido aprobados, se aplicarán los procedimientos de software ya definidos para esta tarea y como parte de la finalización del proceso de cambios, estos deben ser expuestos a auditorías de configuración y verificaciones de calidad, con el objetivo de asegurar que se ha modificado únicamente lo aprobado en el requerimiento y que su funcionalidad es la óptima esperada; finalmente producto de estas auditorías y verificaciones se deben emitir informes que se adjunten como parte del proceso.

Adicionalmente y como parte del proceso de Control de Cambios SWEBOK plantea desviaciones y remisiones, las cuales son una autorización ya sea para abandonar un requerimiento antes del desarrollo del elemento o la autorización para utilizar un elemento ya existente de alguna manera específica con el objetivo de cubrir de cierta forma el requerimiento; en cualquiera de los dos casos se requiere de un procedimiento formal, justificado y autorizado.

3.5 Estado del arte de las herramientas de administración de cambios

El uso de herramientas puede ayudar a que el proceso de solicitud de cambios se ejecute de manera más eficiente; la automatización nos puede apoyar para iniciar solicitudes de cambio, hacer cumplir el flujo del proceso de cambios, capturar las decisiones del comité de control de cambios y obtener reportes acerca de la información del proceso. Sin embargo, no hay que olvidar que una herramienta no es un proceso. Emplear una herramienta para administrar los cambios de software nunca reemplazará a un proceso bien definido, que describa el contenido y la forma de procesar las solicitudes de cambio.

Para obtener un panorama más real sobre el tipo de funcionalidad que proveen las herramientas de administración de cambio, se han seleccionado tres de ellas para conocer su funcionalidad y determinar cuál es la más conveniente para aplicar dentro de Softcase: IBM RationalClearQuest, Atlassian JIRA y Microsoft TeamFoundation Server.

[10]IBM RationalClearQuest

Desarrollado por Rational Software, el cual fue comprado por IBM en el año 2003, su última versión estable es la 8.0.1.9 liberada el 16 de septiembre del 2015.

Es un sistema de producción y desarrollo de aplicaciones de flujo de trabajo que permite personalización dependiendo de las necesidades de la organización.

Proporciona una gran ayuda para administrar de manera efectiva el ciclo de vida, al automatizar los flujos de trabajo, permite controlarlos y hacer cumplir los procesos de desarrollo establecidos desde la definición de un requisito hasta su implementación.

RationalClearQuest, provee de un adecuado seguimiento de cambios y defectos, procesos personalizables, generación de informes tiempo real y rastreabilidad del ciclo de vida para obtener una mayor visibilidad y control del ciclo de vida de desarrollo de software. IBM RationalClearQuest proporciona soporte multi-plataforma escalable a organizaciones de cualquier tamaño, para que pueda seguir personalizando procesos a medida que los requisitos de desarrollo evolucionan.

IBM RationalClearQuest, brinda colaboración a las organizaciones para:

Mejorar la calidad del software con prestaciones incorporadas de seguimiento de defectos y cambios

- Mejora la visibilidad y el control sobre los proyectos con aplicación de procesos y creación de informes consolidada, prácticamente en tiempo real.
- Facilita una gestión de cambios más efectiva con mejoras de rendimiento, usabilidad, integración y seguridad.
- Ayuda a eliminar errores de software que se producen debido a procesos manuales y transferencias, mala comunicación e información incoherente.

[10] [17] [18]

- Orienta a los miembros del equipo ya que las consultas e informes permiten ver las asociaciones de requisitos y el estado de las actividades de planificación de pruebas, autoría de pruebas, ejecución de pruebas, etc., teniendo así un panorama general del estado de cada caso.

Personalizar y automatizar los flujos de trabajo para alcanzar una mejor eficacia y capacidad de predicción

- Ayuda a controlar y aplicar los procesos de desarrollo, desde la definición de requisitos hasta la producción, a fin de mejorar la coordinación y la comunicación del equipo.
- Agiliza el proceso que dirige ciclos de pruebas más rápidos y frecuentes, permitiéndole detectar errores en etapas tempranas del ciclo de entrega de software.
- Mejora la comunicación y la coordinación del equipo con flujos de trabajo automatizados
- Las notificaciones por correo electrónico ayudan a garantizar que se avise a los miembros del equipo correspondientes cuando se requiere su intervención, mejorando los tiempos de respuesta
- Brinda control sobre los procesos, asegurando así que cada requerimiento realizado sea atendido, reduciendo así el riesgo del proyecto.

Simplificar la gestión de la conformidad con herramientas que ayudan a gestionar de forma eficaz los procesos de conformidad y a realizar el seguimiento de las aprobaciones

- Le permite visualizar las asociaciones de requisitos y el estado de sus actividades de planificación, creación y ejecución de pruebas, utilizando consultas e informes.

- Amplía la rastreabilidad en todo el ciclo de vida de entrega de software vinculando requisitos, código, registros de compilación, storyboards, registros de despliegue y otros activos de desarrollo.
- Gestiona todo el rango de actividades de pruebas, desde la planificación hasta la ejecución, la captura y el análisis de resultados de pruebas.

Obtener visibilidad sobre los proyectos con informes prácticamente en tiempo real para mejorar la toma de decisiones

- Le proporciona información sobre sus procesos con soporte a la generación de consultas, gráficas e informes. Las gráficas de distribución, tendencias y envejecimiento le ayudan a visualizar datos complejos.
- Los gráficos pueden ser fácilmente creados y reajustados para permitirle profundizar en el área de datos que se requiera.
- Da soporte a miles de usuarios que trabajan en distintos sitios.

El costo actual de la licencia deIBM RationalClearQuest por Usuario Autorizado, Licencia Inicial + Suscripción y Soporte 12 Meses, tiene un costo de $1,287.79 cada unidad, considerando que Softcase cuenta con un staff de 8 desarrolladores, la inversión total sería de $10,302.32.

[11]Atlassian JIRA

Desarrollado por la empresa australiana Atlassian y lanzado por primera vez el 12 de octubre del 2004, su última versión estable es la 6.1 liberada el 24 de septiembre del 2013.

Es una aplicación basada en web para el control y seguimiento de errores e incidentes y para la gestión operativa de proyectos, se usa también en áreas no técnicas para la administración de tareas. Fue desarrollada inicialmente para el desarrollo de software, sirviendo de apoyo para la gestión de requisitos, seguimiento del estatus y finalmente para el seguimiento de errores. JIRA puede ser utilizado para la gestión de procesos y para la mejora de procesos, gracias a sus funciones para la organización de flujos de trabajo. Permite planificar, supervisar y publicar software de óptima calidad acortando los tiempos y evitando duplicidad de trabajo por falta de comunicación o seguimiento de tareas.

Ayuda a los usuarios de diversas maneras, atacando las distintas áreas del proyecto:

- Permite Planificar, creando casos de usuario e incidencias, planifica sprints y distribuye tareas entre los miembros del equipo de software.
- Permite Supervisar, priorizando y analizando el trabajo del equipo de trabajo en su contexto y con total visibilidad.
- Permite Implementar con confianza y seguridad, sabiendo que la información con la que se cuenta es siempre la más actualizada.
- Permite crear informes, para tomar decisiones y mejorar el rendimiento del equipo con datos visuales en tiempo real de gran utilidad.

Dentro de sus funcionalidades están:

[1111] [19] [20]

- Permite seleccionar flujos de trabajo predefinidos o crear flujos completamente personalizados que se adaptan a la metodología de trabajo de la empresa.
- Proporciona Tableros de Scrum personalizables, mediante los cuales los miembros del equipo pueden mantenerse centrados en cada una de las actividades que se están desarrollando y el estado de las mismas.
- Cuenta con tableros de Kanban flexibles, los cuales proporcionan una total visibilidad respecto a las tareas futuras que permiten rastrear y organizar el trabajo para lograr la máxima producción en ciclos de mínima duración, gestionando y midiendo los flujos de trabajo.
- Permite acceder a más de una decena de reportes pre definidos que recopilan información sobre los sprints realizados.
- JIRA presenta dos opciones de almacenamiento, en la nube (JIRA Software Cloud) o en servidores propios (JIRA Software Server o JIRA Software Data Center).

Los beneficios que se obtienen al usar JIRA son que al ser un software simple, poderoso y amigable, permite al usuario administrar las tareas, defectos, trámites, requerimientos o procesos de forma ágil y adecuada, permite adjuntar documentos que soporten decisiones, cuenta con un excelente sistema de búsqueda de actividades en lenguaje natural, es compatible con casi todas las bases de datos y es de fácil extensión e integración con otros sistemas.

El costo actual de la licencia de JIRA Software Cloud es de $100 anuales de 1-10 usuarios, incluye soporte de 12 Meses, considerando que Softcase cuenta con un staff de 8 desarrolladores, el costo total de inversión seria $100 por año.

[12]Microsoft TeamFoundation Server (TFS)

Desarrollado por Microsoft, incluido dentro del paquete de Microsoft Visual Studio a partir de la versión 2005 lanzada en el año 2006, su última versión estable es la 2015 Update 3 liberada el 27 de junio del 2016.

Es un grupo de tecnologías y herramientas que permiten a los miembros de un equipo colaborar y coordinar sus esfuerzos a la hora de crear un producto o llevar a cabo un proyecto, mejora la comunicación entre los miembros y realiza un seguimiento del estado de las tareas que desarrollan.

Ofrece los elementos necesarios para realizar una acercada administración de cambios, realizando seguimientos exhaustivos de errores, tareas o requisitos; gestionando cada uno de ellos de forma individual con sus respectivos flujos de trabajo, atributos y trazabilidad a pesar que su desarrollo no está específicamente dirigido a esta área.

Posee una biblioteca de documentos en la cual cada elemento de trabajo puede ser vinculado a un hipervínculo donde pueden ser accesados en cualquier momento, es muy útil ya que se tiene acceso directo a informes, tareas de desarrollo, requisitos relacionados a las mismas, código fuente modificado, resultados de pruebas de validación, entre otros. Entre los tipos de elementos más comunes proporcionados por TFS están: requisito de error, riesgo, escenario, tarea, etc., sin embargo permite también la creación de elementos personalizados; proporcionando así un nivel de granularidad adecuado para el control de cambios y trazabilidad.

[12] [21] [22] [23]

El objetivo de TeamFondation con respecto a mejorar la comunicación entre miembros de un equipo es evitar pérdida de información o trabajo ya realizado cuando existe rotación de tareas entre los miembros, para ello el proyecto al cual tienen acceso todo el personal que trabaja dentro del mismo es almacenado con un nombre claro y nemotécnico de modo que sea de fácil ubicación y además proporciona una ubicación central para que el trabajo de los usuarios sea perfectamente coordinado.

TeamFoundation cuenta con el envío de alertas mediante correo electrónico para los miembros de los distintos equipos de un proyecto; dichas alertas pueden contener el estado de alguno de los elementos de trabajo, el inicio de un requerimiento, la finalización de un caso, entre otros.

Permite realizar un seguimiento del estado de cada elemento de trabajo, vigilando de cerca el proyecto, observando el avance de cada tarea y el personal asignado para realizar cada una de ellas. Cada elemento de trabajo mantiene un historial de cada actividad realizada, de manera que cualquier persona que lo consulte sepa exactamente qué sucede, como ha evolucionado, los inconvenientes que se han presentado, etc.

Cuenta con filtros avanzados y útiles al momento de realizar consultas con respecto a elementos de trabajo, por ejemplo obtener las tareas programadas con mayor tiempo asignado, consultar el número de tareas asignadas a un responsable, entre otras.

Adicionalmente provee de varios informes que colaboran a la toma de decisiones, como por ejemplo: casos cerrados, promedio de tiempo de demora por tipo de elemento, resultados de pruebas, históricos de proyectos anteriores para compararlos con el actual; y además brinda la posibilidad de crear informes personalizados de acuerdo a las necesidades de la empresa.

El costo actual de la licencia de Microsoft TeamFoundation, licencia Servidor + CAL, es perpetuo y es de $560 dólares por servidor y $487 por cliente, considerando que Softcase cuenta con un staff de 8 desarrolladores, el costo total sería de $4456.

CAPITULO 4
DISEÑO DEL PLAN DE ADMINISTRACION DE CAMBIOS

4. DISEÑO DEL PLAN DE ADMINISTRACION DE CAMBIOS

Este capítulo detalla el Plan de Administración de Cambios, para el cual se han considerado los aspectos más relevantes que dictan las mejores prácticas de los modelos de madurez citados anteriormente; complementado con las necesidades expuestas por el personal de Softcase.

4.1 Organización

El presente plan está organizado de acuerdo a las funciones desempeñadas por el personal de Softcase y al principal rol que cada persona cumple dentro de la empresa, para ello se describirán aspectos de organización de los roles, definiendo cómo un rol puede participar en alguno de los siguientes grupos o áreas establecidas:

- Administración de defectos
- Administración de solicitudes de mejora
- Administración de actividades generales

Dichas áreas fueron definidas tomando en consideración las fuentes más frecuentes de cambio definidas por las buenas prácticas de la Administración de Cambios y en base a los casos de soporte más repetitivos entre el personal de Softcase, donde se pudo observar que se agrupan en esas tres categorías principalmente manejadas dentro de la empresa.

Administración de Defectos

Se definen como Defectos a cualquier tipo de error encontrado dentro del sistema, ya sea que afecte o no a su funcionamiento.

Rol	Funciones
Solicitante	Describe el defecto encontrado, detalla su procedencia y envía la información.
Coordinador de Cambios	Asigna el caso para ser solucionado o probado; además se encarga de cerrarlo cuando ha sido liberado.
Desarrollador	Corrige el defecto.
Integrador	Integra a una línea base los artefactos que fueron agregados, modificados o eliminados producto de la corrección del defecto.
Tester	Garantiza que la solución implementada cumpla satisfactoriamente todos los casos de prueba que pueda aplicar en relación al requerimiento.

Tabla 2: Roles y funciones para la administración de defectos

El diagrama de actividad que muestra la interacción de los roles más relevantes dentro del proceso y las funciones principales descritas en la Administración de defectos desde su recepción hasta su solución es el siguiente:

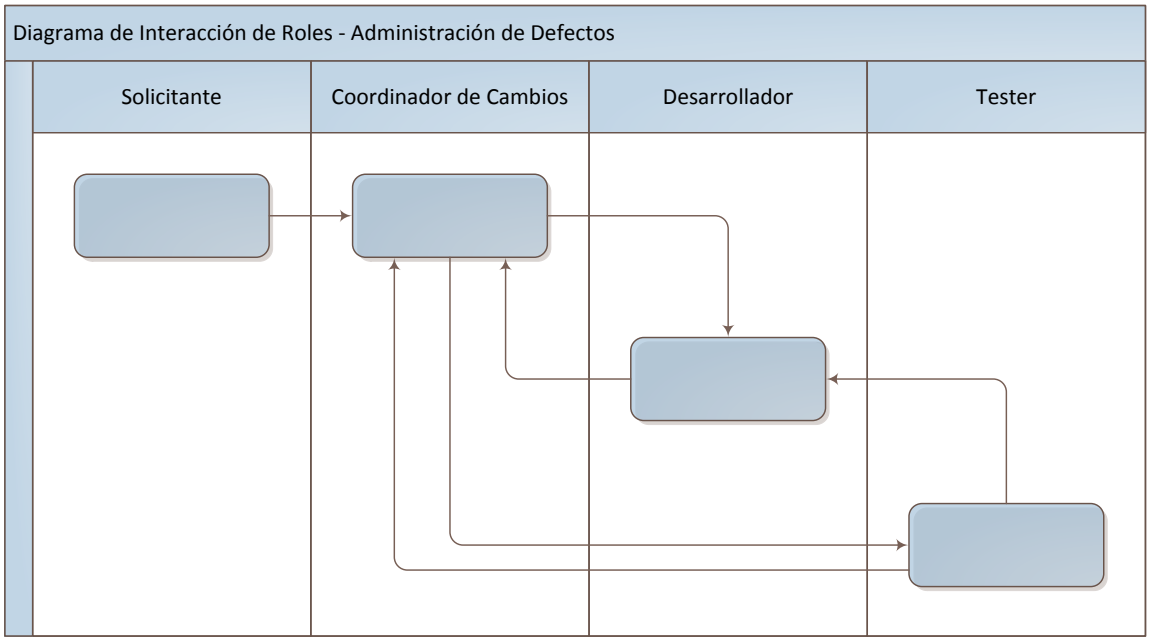


Ilustración 7: Diagrama de actividad de roles para la administración de defectos

Administración de Solicitudes de Mejora

Se definen como Solicitudes de Mejora a cualquier requisito para optimizar el funcionamiento del sistema.

Rol	Funciones
Solicitante	Describe la mejora requerida y envía la información.
Comité de Control de Cambios	Recibe, analiza y asigna a los Administradores de Proyecto, los requerimientos de mejora que consideren viables.
Administrador de Proyecto	Define y asigna las actividades necesarias para implementar la solicitud de mejora.
Coordinador de Cambios	Asigna las pruebas de las solicitudes de mejora que hayan sido implementadas; además se encarga de cerrarlas cuando hayan sido liberadas.
Desarrollador	Realiza el desarrollo de cada una de las actividades definidas y asignadas por el Administrador del Proyecto.
Integrador	Integra a una línea base las actividades realizadas para cumplir la solicitud de mejora.
Tester	Garantiza que la solicitud de mejora implementada cumpla satisfactoriamente todos los casos de prueba que pueda aplicar en relación al requerimiento.

Tabla 3: Roles y funciones para la administración de solicitudes de mejora

El diagrama de actividad que muestra interacción de los roles más relevantes dentro del proceso y las funciones principales descritas en la Administración de solicitudes de mejora desde su recepción hasta su solución es el siguiente:

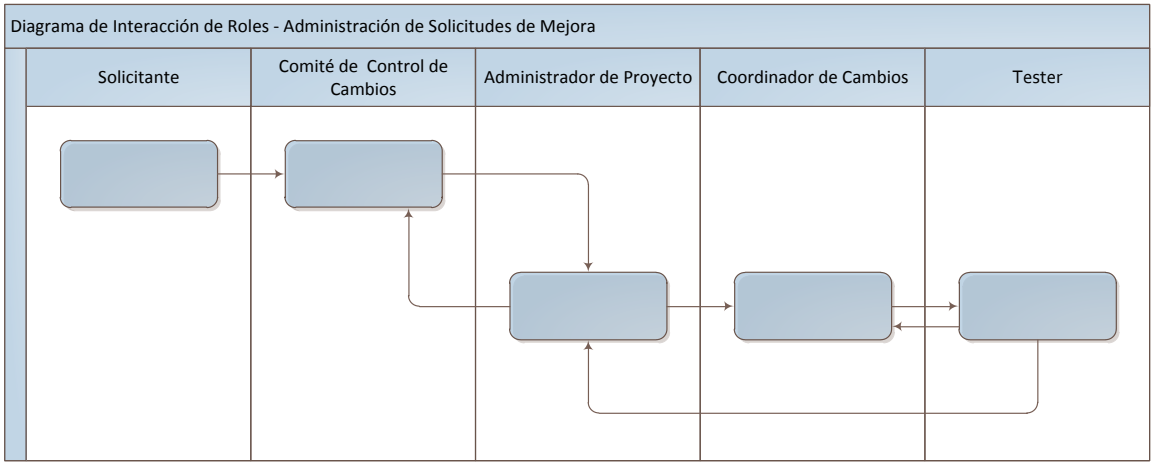


Ilustración 8: Diagrama de actividades para la administración de solicitudes de mejora

Administración de Actividades Generales

Se definen como actividades generales a cualquier actividad que deba ser implementada en un proyecto en marcha.

Rol	Funciones
Solicitante	Envía la información de la actividad que será desarrollada.
Administrador de Proyecto	Define y asigna las actividades necesarias para implementar el proyecto.
Desarrollador	Desarrolla cada una de las actividades definidas y asignadas por el Administrador de Proyecto.
Integrador	Integra a una línea base las actividades realizadas para que sean probadas.
Tester	Garantiza que las actividades implementadas cumplan satisfactoriamente todos los casos de prueba que pueda aplicar en relación al requerimiento.

Tabla 4: Roles y funciones para la administración de actividades generales

El diagrama de actividad que muestra la interacción de los roles más relevantes dentro del proceso y las funciones principales descritas en la Administración de actividades generales de un proyecto, desde su recepción hasta su ejecución es el siguiente:

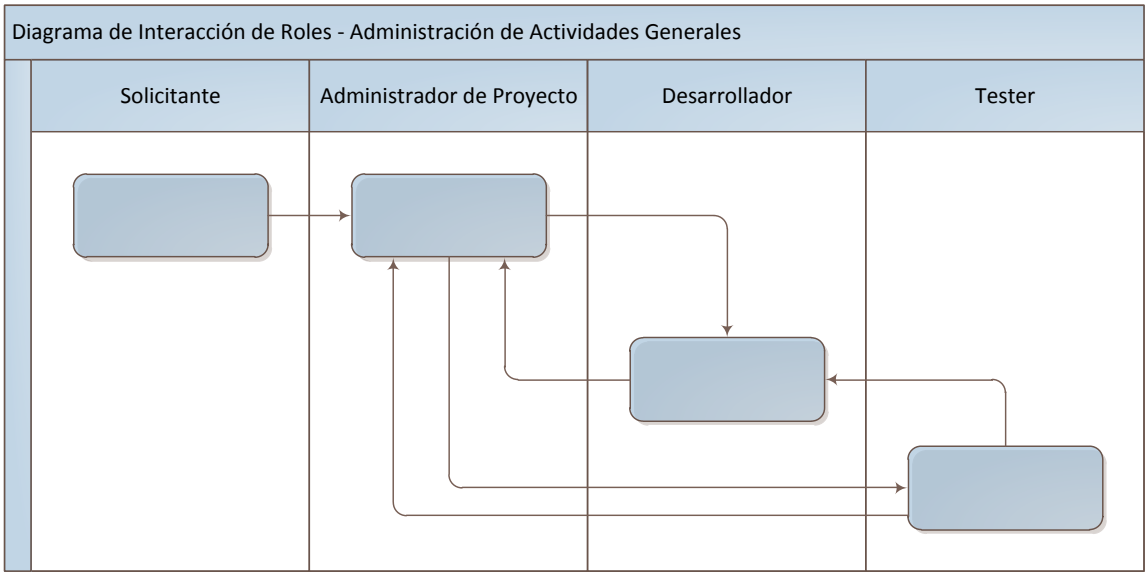


Ilustración 9: Diagrama de actividades para la administración de actividades generales

4.2 Responsabilidades

Cada miembro de la empresa cumple con uno o más roles de los involucrados en las administraciones anteriormente descritas que les ha sido asignado, el cual incluye ciertas responsabilidades que se describen a continuación:

Solicitante

Es cualquier persona que envíe una solicitud de cambio (defecto, petición de mejora, actividades generales en proyectos en marcha) para que sea atendido por Softcase.

Comité de Control de Cambios

Está integrado por el Jefe y el Coordinador del Departamento de Tecnologías, se encargan de evaluar si las solicitudes de cambio o mejora deben ser aprobadas o rechazadas; y en caso de ser aprobadas son los encargados de asignar dichas solicitudes a los Administradores de Proyecto correspondientes.

Administrador de Proyecto

Todas las personas que tengan asignado el rol de Administrador de Proyecto son candidatas a recibir la asignación de una solicitud de cambio o mejora que previamente haya sido aprobada por el Comité de Control de Cambios.

Se encarga de crear y asignar las actividades necesarias para lograr implementar la solución al requerimiento que le fue asignado.

Además, tiene la responsabilidad de asegurar que los estándares del proyecto sean correctamente aplicados y verificados por todos los miembros del mismo.

Desarrollador

Es el encargado de desarrollar las actividades necesarias para atender: una solicitud de mejora, la corrección de un defecto o una actividad de un proyecto en marcha.

Coordinador de Cambios

Es el responsable de realizar la asignación de:

- Defectos que han sido enviados pero que aún no han sido asignados a un Desarrollador para que sea resuelto.
- Defectos que han sido resueltos y que están pendientes de ser probados.

- Solicitudes de mejora que han sido implementadas y que están pendientes de ser probadas.

Además es responsable de efectuar el cierre definitivo de cada caso abierto ya sea por defectos o solicitudes de mejora, que estén probados y han sido liberados a producción exitosamente.

Tester

Este rol será ejecutado por cualquier miembro del departamento de pruebas y soporte, su responsabilidad es definir y ejecutar casos de prueba que validen la solución implementada por el desarrollador. El tester podrá devolver la solución propuesta cuando ésta no cumpla satisfactoriamente con las pruebas realizadas, o por el contrario podrá certificar su correcto funcionamiento para ser liberada a producción.

Integrador

Cuando los desarrolladores entregan sus componentes probados, los integradores son los encargados de combinarlos para producir una versión operativa de un módulo que operará dentro del ERP de Softcase.

Su responsabilidad es planear la integración a nivel de sistemas y subsistemas, cada uno en espacios de trabajo separados.

Administrador de la Configuración

Es el encargado de proveer a los desarrolladores e integradores áreas de trabajo apropiadas para construir y probar su trabajo, además de que todos los artefactos estén siempre disponibles para incluirlos en las unidades de puesta en marcha según se requiera.

Facilita las actividades de revisión de productos, cambios y seguimiento de defectos.

Y finalmente es el responsable de mantener este plan y reportar las estadísticas basadas en métricas previamente establecidas con respecto a las solicitudes de cambio que se presenten, de manera que aporten como una guía a la empresa para la toma de decisiones.

4.3 Herramientas

Luego de hacer una breve revisión de tres herramientas disponibles en el mercado para manejar una correcta Administración de Cambios, se analizó si cumplían con las necesidades de Softcase, obteniendo la siguiente tabla comparativa:

Necesidades a Cubrir	IBM RationalClearQuest	Atlassian JIRA	Microsoft TeamFoundation Server (TFS)
Fácil de usar	✓	✓	✓
Conveniente Económicamente		✓	✓
Almacenamiento en la nube sin necesidad de otro componente	✓	✓	
Parametrizable	✓	✓	✓
Reportes estadísticos personalizables	✓	✓	✓
Disponibilidad de Soporte	✓	✓	✓

Tabla 5: Matriz de análisis de herramientas en base a requerimientos

Como se puede observar las tres herramientas cumplen con la mayoría de características que Softcase requiere.

Desde el punto de vista económico, tanto JIRA ($100 anual) como TeamFoundation Server ($4456 perpetuo), resultan una mejor alternativa con respecto a RationalClearQuest.

Sin embargo JIRA además, brinda la alternativa de almacenamiento en la nube y considerando que Softcase también busca asegurar la disponibilidad y el acceso total a

su información desde cualquier punto donde su personal se encuentre laborando, sería la opción más adecuada.

Considerando los aspectos anteriormente mencionados y para dar cumplimiento al objetivo de recomendar una de ellas para la posterior implementación del Plan de Administración de Cambios que será definido; se recomienda el uso de la herramienta JIRA de la compañía Atlassian, debido a las necesidades, al número de personas que necesitarían acceso a la herramienta y al presupuesto de Softcase.

En caso que Softcase opte por la implementación del Plan de Administración de Cambios en JIRA, deberá realizar las siguientes actividades:

- Adquirir el paquete básico de licencias de JIRA, al menos inicialmente hasta que se evalúe si es necesario ampliar el número de licencias de cliente
- Asegurarse de contar con una infraestructura que soporte los requerimientos básicos de JIRA
- Brindar la capacitación necesaria al personal de la empresa que vaya a trabajar directamente con la herramienta
- Socializar el Plan de Administración de Cambios
- Implementarlo en la herramienta

El Plan de Administración de Cambios

El objetivo principal de este proyecto es crear un plan de Administración de Cambios, que se adapte a las necesidades de Softcase, para lo cual en las secciones siguientes, se identificará, diseñará y documentará un enfoque sistémico para procesar las solicitudes de corrección de defectos, solicitudes de mejora y actividades generales que se presenten dentro de un proyecto de software de cualquier cliente de la empresa.

Para cada tipo de solicitud ya sea corrección de defecto, solicitud de mejora o solicitud de una actividad general, se identificarán cuatro elementos que permitirán definir una guía adecuada para su automatización posterior.

Estos elementos son:

- **Campos y estructura.** Como su nombre lo indica, este elemento define la información que debe levantarse al momento de conformar un registro de requerimiento, este registro sería el insumo necesario para el diseño de los formularios en caso de que el Plan sea automatizado.
- **Flujo de trabajo.** Contiene todas las posibles transiciones y estados por los que podría pasar una solicitud desde su envío hasta su cierre formal.
- **Acciones.** Se indican las acciones válidas en un momento específico del proceso para cada solicitud (defecto, mejora, actividad general); para cada acción se indica el rol habilitado para ejecutarla y sus excepciones de ejecución en caso de existir
- **Comportamientos:** Indica para cada campo, de cada solicitud, en un estado determinado; si es mandatorio u opcional.

En los siguientes apartados se describirán cada uno de los elementos citados anteriormente, para cada tipo de solicitud.

4.4 Administración de Defectos

4.4.1 Campos y estructura

Para cada corrección de defecto que se recepte, deberán registrarse los siguientes campos:

Campo	Descripción
Cliente	Cliente que envía el Defecto encontrado.
Ambiente	Indica el ambiente de trabajo en donde se detectó el defecto, puede ser: *Desarrollo:* Ambiente en el cual se desarrollan las aplicaciones, si un defecto es encontrado aquí por un desarrollador es cuando aún no ha salido a producción. *Pruebas:* Ambiente prácticamente igual al de producción, en donde los tester aplican todo tipo de casos de prueba para validar el componente antes de ser liberado a producción, es aquí donde comúnmente son detectados los defectos. *Producción:* Ambiente en donde los usuarios finales ejecutan las aplicaciones, si un defecto es encontrado en este ambiente, será reportado por un usuario.
Aplicativo	Ruta completa de acceso del aplicativo donde fue detectado el defecto: Subsistema\Menú\Submenú\Opción\módulo/reporte
Descripción	Descripción detallada del defecto encontrado.
Archivos	Archivos que ayuden a evidenciar el hallazgo, capturas de pantalla, reportes, etc.
Detector	Persona que detecta el defecto.
Comunicador	Persona que comunica el defecto.
Email contacto	Correo electrónico al que debe comunicarse cualquier información

	con respecto al caso.
Estado	Estado actual del defecto.
Fecha de detección	Fecha en la que se detecta el defecto.
Fecha de envío	Fecha en la que se envía la solicitud de corrección de defectos.
Fecha de inicio	Fecha en la que se inicia a trabajar en la solución.
Fecha de cierre	Fecha en la que se cierra el caso.
Duración	Tiempo que tomó en resolver el defecto.
Fecha de integración	Fecha en la que se libera la solución en ambiente de producción.
ID Caso	Secuencial que identifica al defecto reportado.
Coordinador de Cambios	Personal que receptó la solicitud de corrección de defecto, quien se encargará de asignar la resolución a un desarrollador y las pruebas a un tester.
Prioridad	Prioridad asignada por el Coordinador de Cambios luego de analizar la solicitud: *Urgente.* Deberá ser resuelto inmediatamente, posponiendo cualquier actividad que se esté desarrollando actualmente y que contenga una distinta prioridad, asignando el 100% del tiempo de trabajo. *Alta.* Deberá ser tratado con premura, inmediatamente después de culminar lo que actualmente se está desarrollando o asignarle el 75% del tiempo de trabajo. *Media.* Deberá ser agendado para su tratamiento, se asignará aproximadamente un 40% del tiempo de trabajo. *Baja.* Deberá ser tratado asignando no más del 20% del tiempo de trabajo.

Desarrollador	Personal del equipo de desarrollo al cual el Coordinador de Cambios asignó la solicitud.
Tester	Personal del equipo de pruebas al cual el Coordinador de Cambios asignó la validación de la solución desarrollada para la corrección del defecto.
Integrador	Personal al cual el Coordinador de Cambios asigna la tarea de implementar en ambiente de producción, la solución previamente validada.
Artefactos	Todos los componentes ya sean modulares o de base de datos que se crearon, modificaron o eliminaron para realizar la corrección al defecto, por ejemplo: Módulos, reportes, scripts ejecutados, objetos de bases de datos, etc. Deberán incluir los siguientes datos: Tipo: modulo, reporte, disparador, paquete, etc. Nombre: nombre del artefacto Fecha de actualización: fecha de última modificación
Script de integración	Pasos a seguir para la implementación de la solución en ambiente de producción, debe incluirse todas las actividades y su orden específico, si la solución incluye objetos de base de datos debe conformarse un script e indicar en qué momento y circunstancia de la integración debe ser ejecutado. Por ejemplo: 1. Copiar modulo “XX” 2. Ejecutar script “YY” 3. Etc.
Registros de Pruebas	Descripción del conjunto de pruebas aplicadas para validar la corrección del defecto

Observaciones	Cualquier observación o nota que sea necesario incluir con respecto a la solicitud de corrección del defecto, deberá incluir: Personal/Cargo: Nombre del personal que escribe la observación y su cargo Notas: Observación Fecha: Fecha en la que realiza la observación

Tabla 6: Campos de la solicitud para corrección de defectos

4.4.2 Flujo de trabajo

Los estados por los que puede atravesar un defecto durante el proceso de su resolución se explicarán usando tres diagramas, donde se evidencian los posibles escenarios que pueden presentarse, para ello a continuación se presenta un listado de los estados probables a lo largo del proceso.

Estado	Descripción
Enviado	Estado inicial del defecto.
Duplicado	Cuando el Coordinador de Cambios detecta que el requerimiento ya ha sido receptado anteriormente.
Asignado	Cuando la solicitud de corrección de defecto ha sido asignada a un Desarrollador.
Abierto	Cuando el desarrollador comienza a trabajar en la solución.
Rechazado	Cuando el desarrollador no logra reproducir el defecto reportado.
Pospuesto	Cuando el desarrollador recibe la orden del Coordinador de Cambios de posponer el trabajo para dedicarse a otra actividad de mayor prioridad.
En Pruebas	Cuando el Tester está aplicando los casos de prueba para validar la solución desarrollada.
Devuelto	Cuando el Tester determina que la solución desarrollada aun presenta fallas y la devuelve al desarrollador para su depuración.

Probado	Cuando la solución ha sido validada satisfactoriamente por el Tester.
Resuelto	Cuando el integrador ha completado la implementación de la solución al defecto.
Cerrado	Cuando la solución ha sido liberada al ambiente de producción, es decir cuando la solución se ha completado exitosamente.

Tabla 7: Estados posibles de una solicitud de corrección de defectos

Diagrama Básico. Define los estados y transiciones normales por las que pasaría un defecto, suponiendo un escenario ideal en donde no se registre ningún inconveniente, es decir el flujo básico.

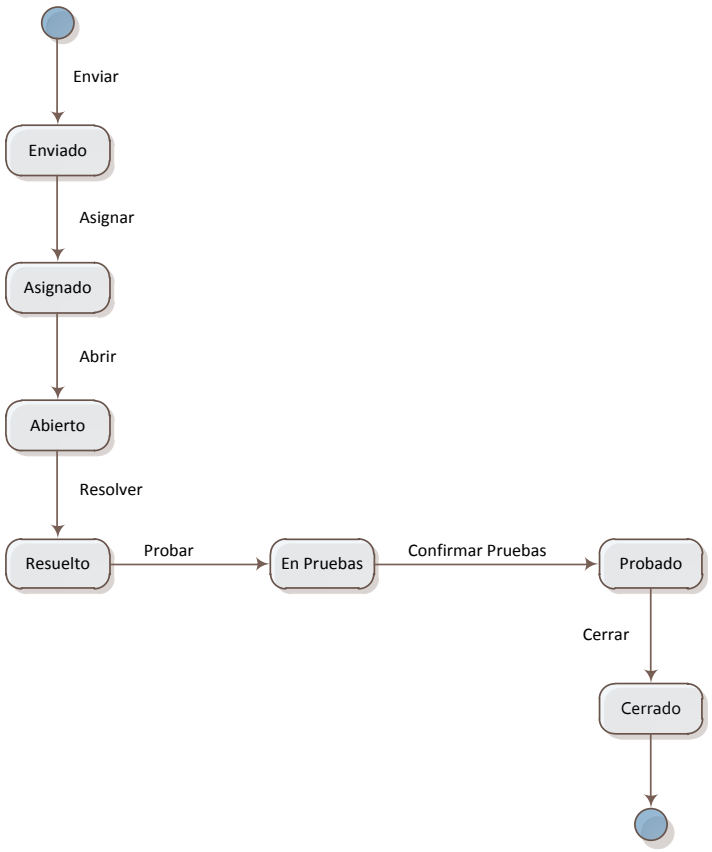


Ilustración 10: Diagrama de flujo básico de una solicitud de corrección de defectos

Diagrama Completo. Este diagrama modela todas las transiciones del diagrama básico anteriormente definido, más todas las posibles transiciones por las que podría pasar un defecto en casos excepcionales.

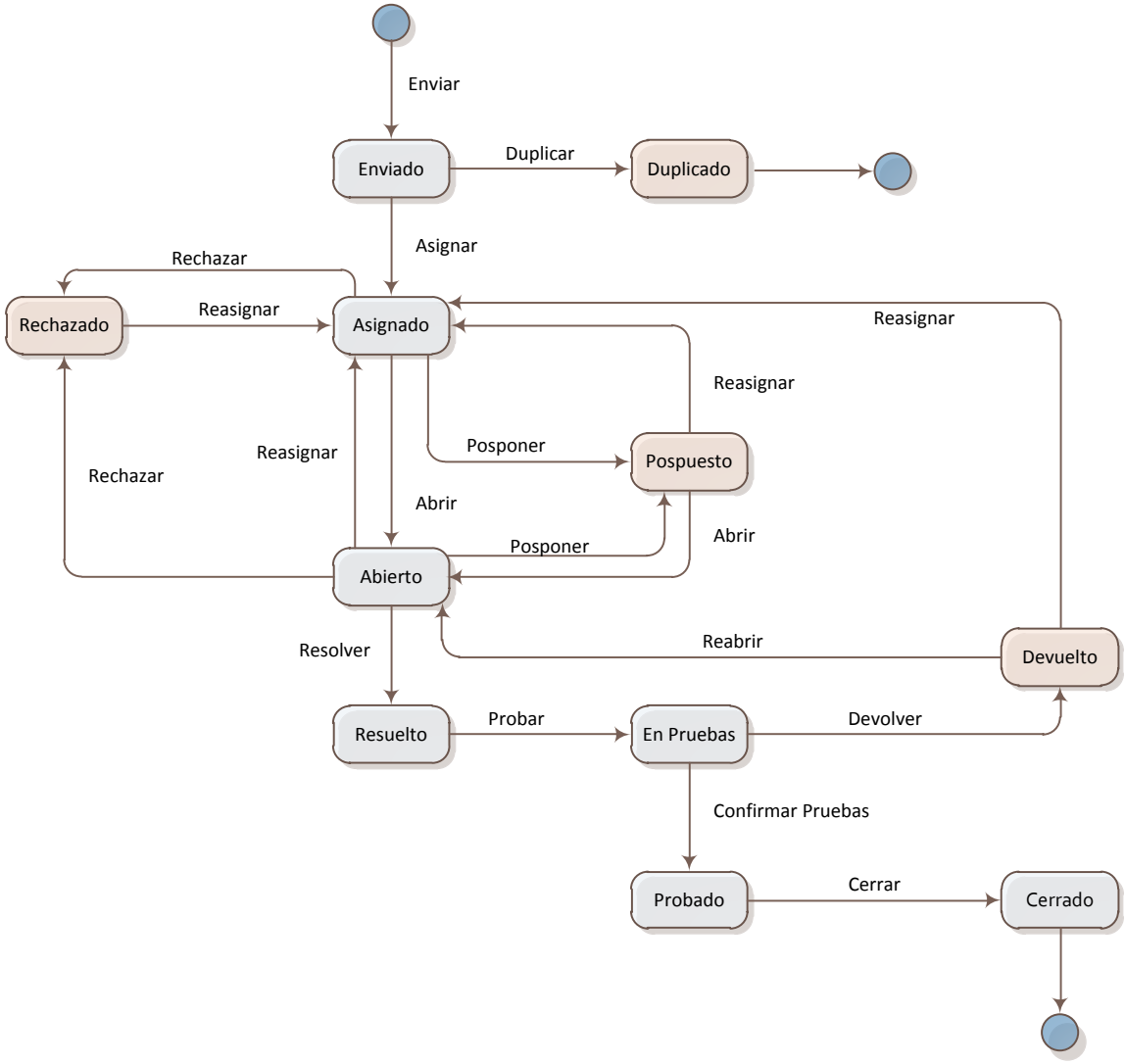


Ilustración 11: Diagrama de flujo completo de una solicitud de corrección de defectos

El siguiente diagrama muestra un ejemplo para una mejor comprensión del flujo completo de transiciones, de una solicitud corrección de defectos.

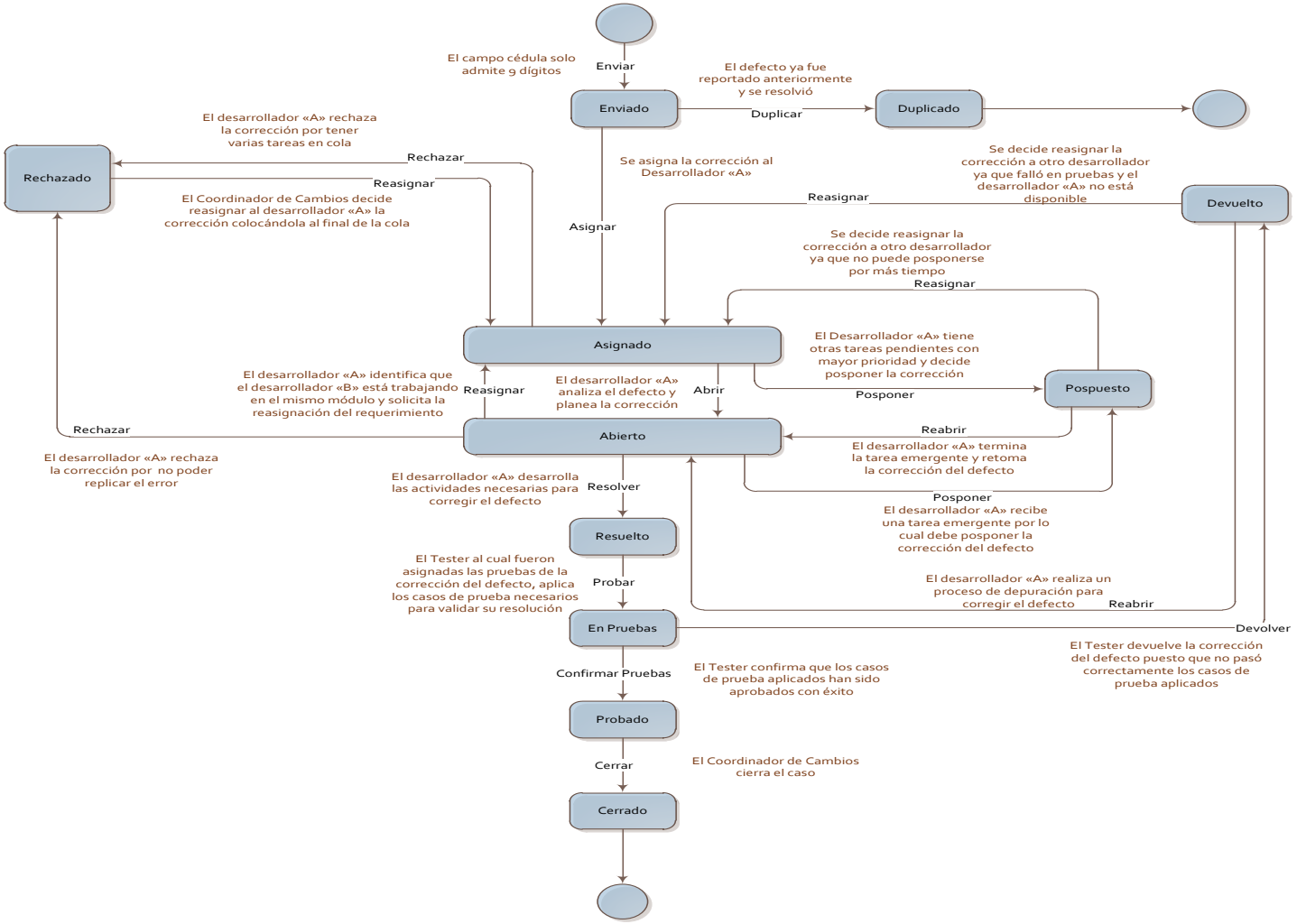


Ilustración 12: Ejemplo de flujo completo de una solicitud de corrección de defecto

Diagrama de Transiciones por roles. Este diagrama muestra la participación de los roles más relevantes durante el ciclo de corrección de un defecto, en todas las posibles transiciones que puedan presentarse.

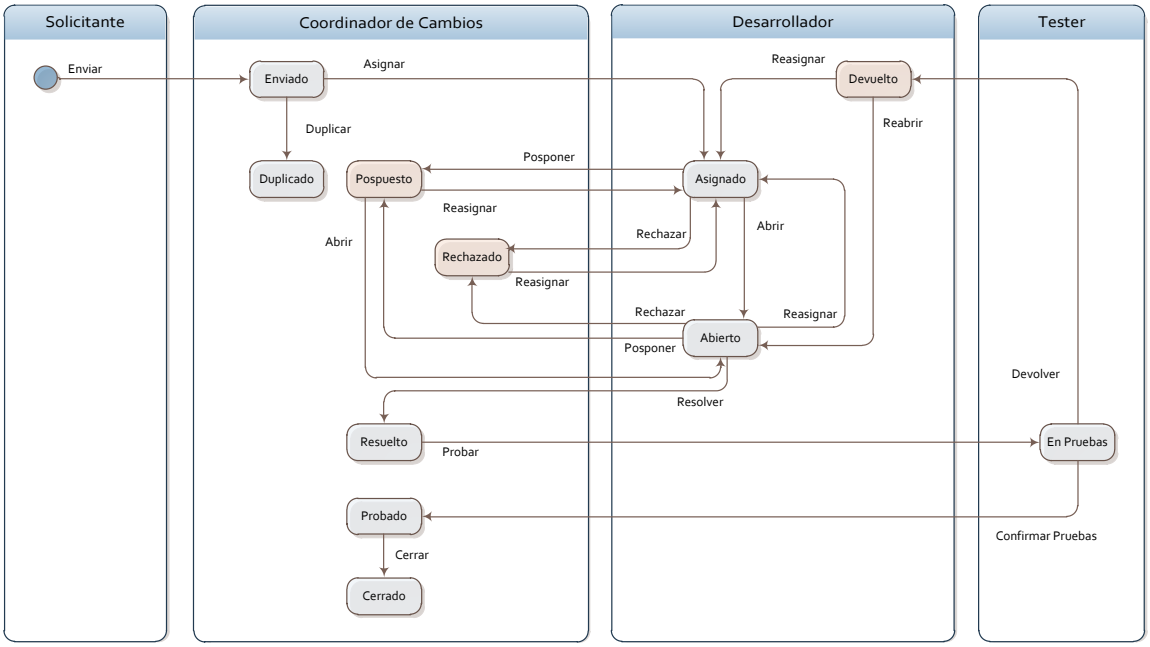


Ilustración 13: Diagrama de transición de actividades por roles para Solicitudes de corrección de defectos

4.4.3 Acciones

Durante todo el proceso de la corrección del defecto, desde su recepción hasta su implementación en ambiente de producción, se presentan distintas acciones las mismas que pueden ser realizadas por ciertos roles específicos en función de sus responsabilidades asignadas.

El siguiente cuadro detalla las acciones que pueden realizarse sobre un defecto y los roles habilitados para ejecutarlas.

Acción	Descripción	Roles Habilitados
Enviar	Reporta una solicitud de corrección de defecto.	• Todos
Asignar	Asigna la corrección del defecto a un Desarrollador.	• Coordinador de Cambios • Comité de Control de Cambios

		• Administrador de Proyecto
Rechazar	Rechaza la asignación de la solicitud de corrección de un defecto por los siguientes motivos: a. Se determinó que no existe el defecto como tal, sino un error de operatividad b. No se logró reproducir el defecto reportado c. El defecto deber ser corregido por otro Desarrollador	• Desarrollador • Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto
Marcar Duplicado	Marca la solicitud de corrección de defecto como duplicada	• Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto
Desmarcar Duplicado	Quita la marca de duplicado de la solicitud de corrección de defectos para que continúe dentro del proceso	• Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto
Abrir	Marca el inicio de los trabajos para corregir el defecto.	• Desarrollador
Resolver	Indica que las tareas de corrección del defecto han concluido.	• Desarrollador
Posponer	Detiene el desarrollo de la solución al defecto	• Desarrollador • Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto
Devolver	Devuelve la solución desarrollada para la	• Tester

	corrección del defecto por no pasar satisfactoriamente las pruebas	• Coordinador de Cambios
Reasignar	Permite reasignar la solicitud de corrección de defecto	• Coordinador de Cambios • Desarrollador (cuando está abierto)
Reabrir	Reabre una solicitud de corrección de defecto cuando ha sido devuelto por alguna razón	• Desarrollador • Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto
Probar	Indica que la solución desarrollada está siendo validada	• Tester • Coordinador de Cambios
Confirmar Pruebas	Indica que las pruebas han culminado satisfactoriamente	• Tester • Coordinador de Cambios
Cerrar	Marca la solicitud como finalizada, es decir que ya fue atendida e implementada al 100% en producción	• Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto

Tabla 8: Acciones y roles para solicitudes de corrección de defectos

4.4.4 Comportamientos

Es de vital importancia definir los comportamientos que puede tener cada campo de las correcciones de defecto receptadas (detallados en la sección 4.4.1) en un estado determinado, puesto que permiten proteger las modificaciones de valores en un punto específico del proceso en el cual no sería óptimo hacerlo.

A continuación se presenta una matriz de posibilidades donde se indica si el cambio es opcional (O), mandatorio (M), o si únicamente es permisible de lectura ( ).

Campo \ Estado	Enviado	Duplicado	Asignado	Abierto	Rechazado	Pospuesto	En Pruebas	Devuelto	Probado	Resuelto	Cerrado
Cliente	M										
Ambiente	M										
Aplicativo	M										
Descripción	M										
Archivos	O	O	O								
Detector	M										
Comunicador	M										
Email contacto	M	O	O	O	O	O	O	O	O	O	O
Estado	M	M	M	M	M	M	M	M	M	M	M
Fecha de detección	M										
Fecha de envío	M										
Fecha de inicio		M		M							
Fecha de cierre		M			M	M					M
Duración		M			M	M					M
Fecha de integración										M	
ID Caso			M								
Coordinador de Cambios			M								
Prioridad			M								
Desarrollador			M								

Tester							M				
Integrador									M		
Artefactos				M							
Script de integración				M					O		
Observaciones	O	O	O	O	O	O	O	O	O	O	O

Tabla 9: Matriz de comportamientos para solicitudes de corrección de defectos

4.5 Administración de Solicitudes de Mejora

4.5.1 Campos y estructura

Para cada solicitud de mejora que se proponga, deberán registrarse los siguientes campos:

Campo	Descripción
Cliente	Cliente que envía la solicitud de mejora.
Aplicativo	Ruta completa de acceso del aplicativo donde desea implementarse la mejora: Subsistema\Menú\Submenú\Opción\
Descripción	Descripción detallada de la mejora solicitada.
Archivos	Archivos que ayuden a clarificar el requerimiento que se presenta.
Solicitante	Usuario que efectuó el envío de la solicitud de mejora.
Email contacto	Correo electrónico al que debe comunicarse cualquier información con respecto al caso.
Estado	Estado actual de la solicitud de mejora.
Fecha de envío	Fecha en la que se envía la solicitud de mejora.
Fecha de inicio	Fecha en la que se inician a desarrollar las actividades para cumplir con la solicitud de mejora.

Fecha de cierre	Fecha en la que se cierra el caso.
Duración	Tiempo que tomó en culminar la solicitud de mejora.
Fecha de integración	Fecha en la que se libera la mejora en ambiente de producción.
ID Caso	Secuencial que identifica a la solicitud de mejora enviada.
Receptor	Miembro del Comité de Control de Cambios que receptó la solicitud de mejora, quien se encargará de asignarla al Administrador de Proyecto correspondiente.
Coordinador de Cambios	Personal que se encargará de asignar las pruebas de la mejora desarrollada a un tester.
Administrador de Proyecto	Nombre del Administrador del Proyecto encargado de planificar, desarrollar y dar seguimiento a la solicitud de mejora.
Prioridad	Prioridad asignada por el Comité de Control de Cambios luego de analizar la solicitud de mejora: *Urgente.* Deberá darse tratamiento a la solicitud inmediatamente, posponiendo cualquier actividad que se esté desarrollando actualmente y que contenga una distinta prioridad, asignando el 100% del tiempo de trabajo. *Alta.* Deberá ser tratado con premura, inmediatamente después de culminar lo que actualmente se está desarrollando o asignarle el 75% del tiempo de trabajo. *Media.* Deberá ser agendado para su tratamiento, se asignará aproximadamente un 40% del tiempo de trabajo. *Baja.* Deberá ser tratado asignando no más del 20% del tiempo de trabajo.

Actividades	Define la lista de actividades de trabajo que se deben desarrollar para completar la petición de mejora.
Desarrollador	Personal del equipo de desarrollo al cual el Administrador de Proyecto asignó la solicitud. Pueden ser más de uno.
Tester	Personal del equipo de pruebas al cual el Coordinador de Cambios asignó la validación de la solución desarrollada.
Integrador	Personal a quien el Coordinador de Cambios asigna la tarea de implementar en producción, la solución previamente validada.
Artefactos	Todos los componentes ya sean modulares o de base de datos que se crearon, modificaron o eliminaron para desarrollar la mejora solicitada, por ejemplo: Módulos, reportes, scripts ejecutados, objetos de bases de datos, etc. Deberán incluir los siguientes datos: Tipo: modulo, reporte, disparador, paquete, etc. Nombre: nombre del artefacto Fecha de actualización: fecha de última modificación
Script de integración	Pasos a seguir para la implementación de la solución en ambiente de producción, debe incluirse todas las actividades y su orden específico, si la solución incluye objetos de base de datos debe conformarse un script e indicar en qué momento y circunstancia de la integración debe ser ejecutado. Por ejemplo: 1. Copiar modulo “XX” 2. Ejecutar script “YY” 3. Etc.
Registros de Pruebas	Descripción del conjunto de pruebas aplicadas para validar que la solicitud de mejora cumple con lo requerido

Observaciones	Cualquier observación o nota que sea necesario incluir con respecto a la solicitud de mejora, deberá incluir: Personal/Cargo: Nombre del personal que escribe la observación y su cargo Notas: Observación Fecha: Fecha en la que realiza la observación

Tabla 10: Campos de la solicitud de mejora

4.5.2 Flujo de trabajo

Los estados por los que puede atravesar una solicitud de mejora durante el proceso de su desarrollo, se explicarán usando tres diagramas, donde se evidencian los posibles escenarios que pueden presentarse, para ello a continuación se presenta un listado de los estados probables a lo largo del proceso.

Estado	Descripción
Enviada	Estado inicial de la solicitud de mejora.
Duplicada	Cuando el miembro del Comité de Control de Cambios que recepta la solicitud de mejora, detecta que el requerimiento ya ha sido receptado anteriormente.
Asignada	Cuando la solicitud de mejora ya ha sido asignada a un Administrador de Proyecto.
Abierta	Cuando el Administrador de Proyecto ha iniciado la planificación de la solicitud de mejora.
Rechazada	Cuando el Comité de Control de Cambios decide que la solicitud de mejora no es viable, para ello se realiza un análisis del impacto que producirían su implementación, a nivel de negocio y de tecnología.
Pospuesta	Cuando el Administrador de Proyecto recibe la orden del Comité de

	Control de Cambios, de posponer el trabajo para dedicarse a otra actividad de mayor prioridad.
Implementada	Cuando la implementación de la petición de mejora se ha completado y está lista para ser enviada a pruebas.
En Pruebas	Cuando el Tester está aplicando los casos de prueba para validar la solución desarrollada.
Devuelta	Cuando el Tester determina que la solución desarrollada presenta fallas o inestabilidad y la devuelve al desarrollador para su depuración.
Probada	Cuando la mejora desarrollada ha sido validada satisfactoriamente por el Tester.
Cerrada	Cuando la solución ha sido liberada al ambiente de producción, es decir cuando la solución se ha completado exitosamente.

Tabla 11: Estados posibles de una solicitud de mejora

Diagrama Básico. Define los estados y transiciones normales por las que pasaría una solicitud de mejora, suponiendo un escenario ideal en donde no se registre ningún inconveniente, es decir el flujo básico.

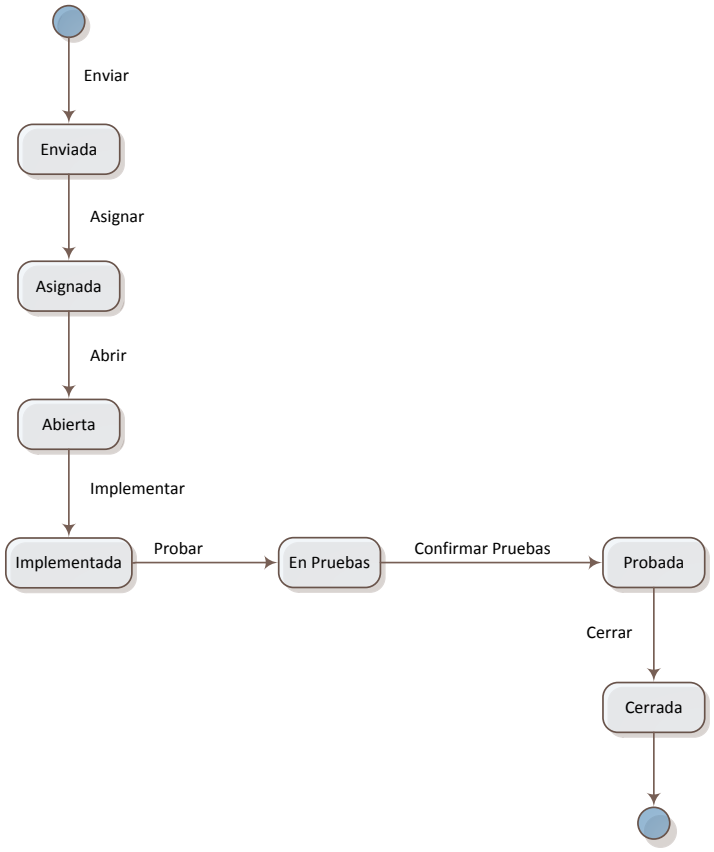


Ilustración 14: Diagrama de flujo básico de una solicitud de mejora

Diagrama Completo. Este diagrama modela todas las transiciones del diagrama básico anteriormente definido, más todas las posibles transiciones por las que podría pasar una solicitud de mejora en casos excepcionales.

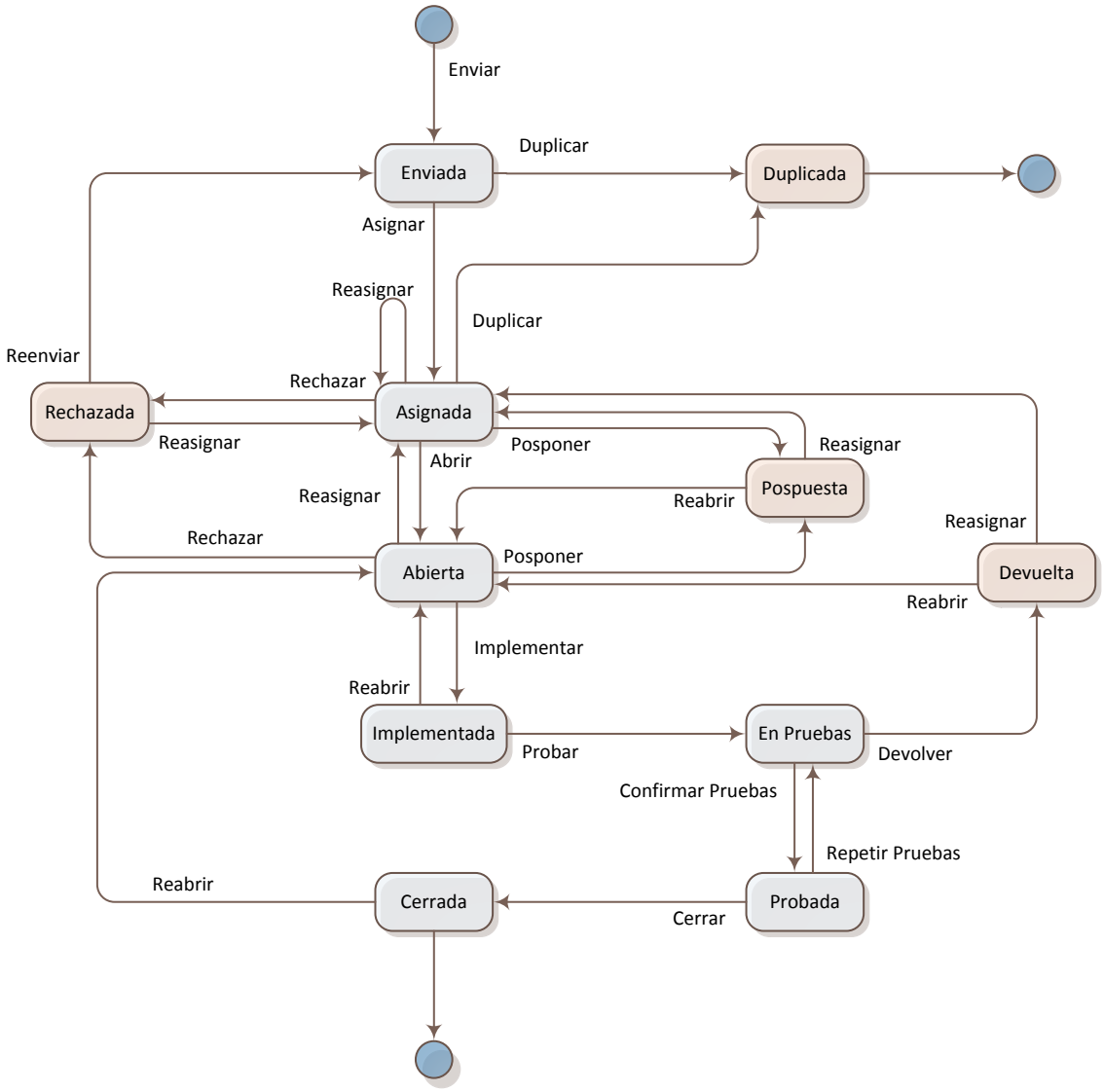


Ilustración 15: Diagrama de flujo completo de una solicitud de mejora

El siguiente diagrama muestra un ejemplo para una mejor comprensión del flujo completo de transiciones, de una solicitud de mejora.

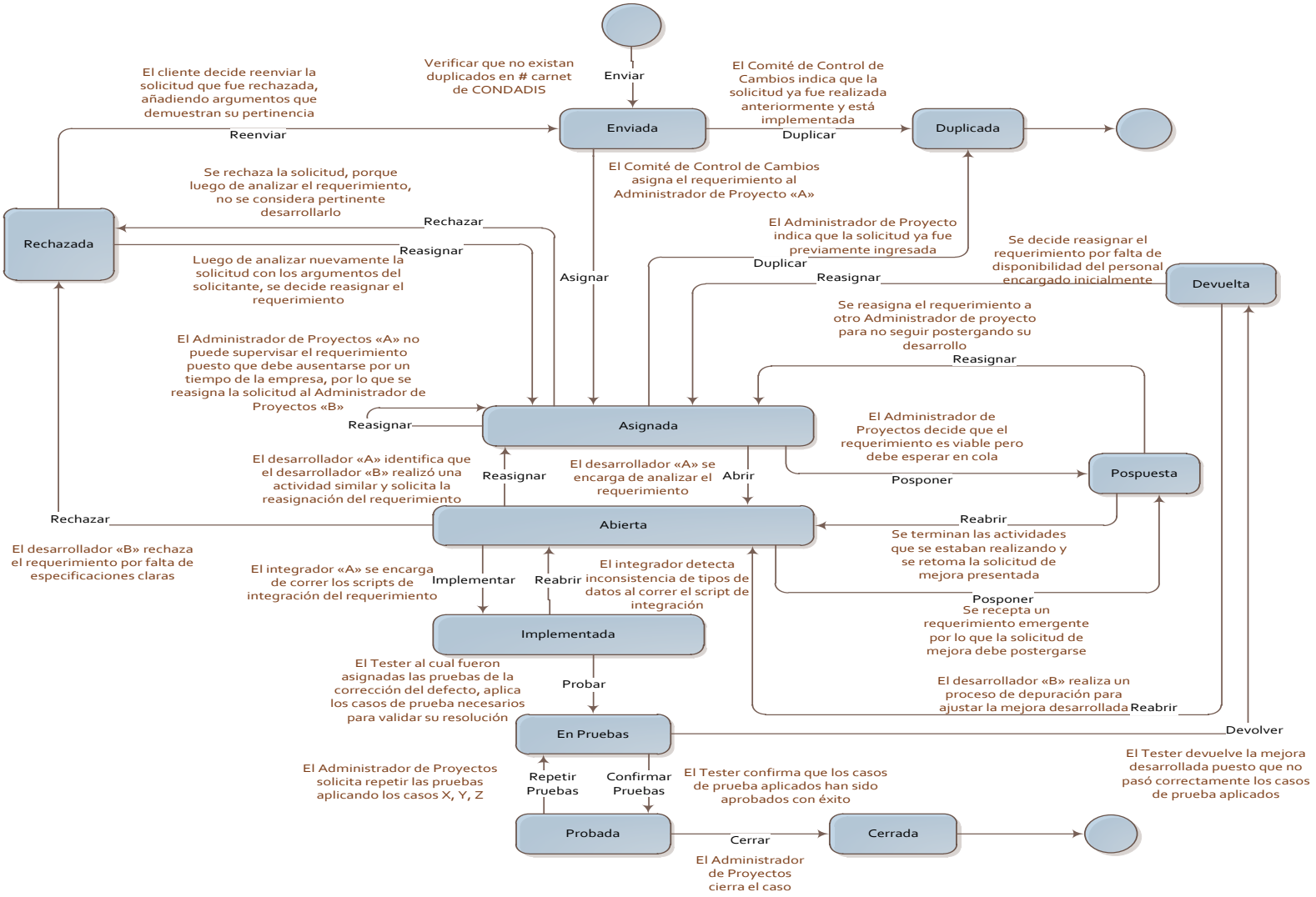


Ilustración 16: Ejemplo de flujo completo de una solicitud de mejora

Diagrama de Transiciones por roles. Este diagrama muestra la participación de los roles más relevantes durante el ciclo que sigue una solicitud de mejora, en todas las posibles transiciones que puedan presentarse.

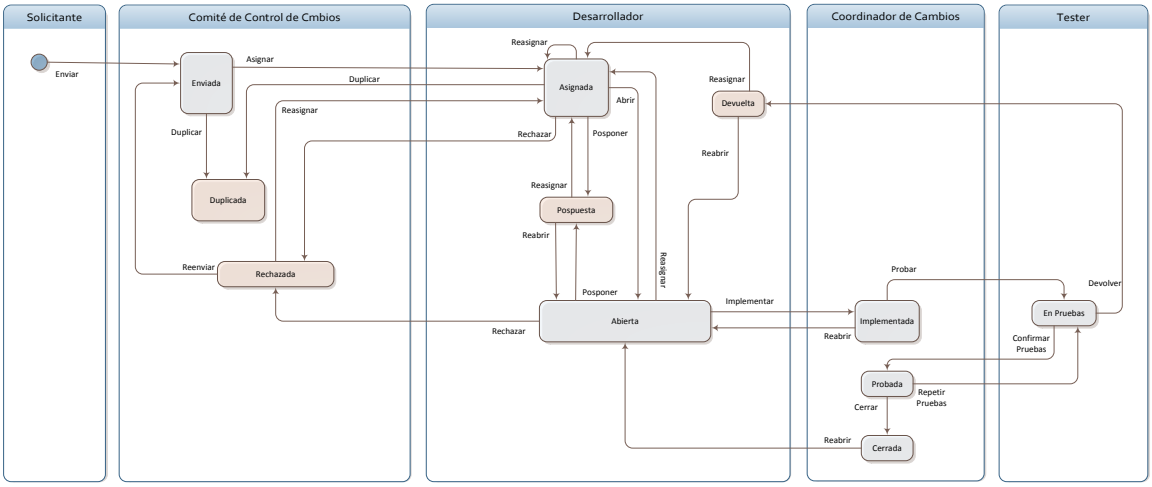


Ilustración 17: Diagrama de transición de actividades por roles para Solicitudes de mejora

4.5.3 Acciones

Durante todo el proceso del desarrollo de la solicitud de mejora, desde su recepción hasta su implementación en ambiente de producción, se presentan distintas acciones las mismas que pueden ser realizadas por ciertos roles específicos en función de sus responsabilidades asignadas.

El siguiente cuadro detalla las acciones que pueden realizarse sobre una solicitud de mejora y los roles habilitados para ejecutarlas.

Acción	Descripción	Roles Habilitados
Enviar	Reporta una solicitud de mejora.	• Todos
Asignar	Asigna la solicitud de mejora a un Administrador de Proyecto.	• Comité de Control de Cambios
Implementar	Indica que se ha implementado la solicitud de mejora.	• Integrador
Rechazar	Rechaza la solicitud de mejora por cualquiera de	Según el caso:

	los siguientes motivos: a) El Comité de Control de Cambios determinó que no es viable b) Fue asignada a un Administrador de Proyecto erróneo	a) Comité de Control de Cambios b) Administrador de Proyecto
Reenviar	Permite efectuar nuevamente el envío de una solicitud de mejora que fue anteriormente rechazada.	• Solicitante
Marcar Duplicado	Marca la solicitud de mejora como duplicada por ya haber sido receptada anteriormente ya sea por el cliente actual o cualquier otro	• Comité de Control de Cambios • Administrador de Proyecto
Desmarcar Duplicado	Quita la marca de duplicado de una solicitud de mejora marcada por error, para que continúe dentro del proceso.	• Comité de Control de Cambios • Administrador de Proyecto
Abrir	Marca el inicio de los trabajos para desarrollar la solicitud de mejora	• Administrador de Proyecto
Posponer	Detiene el desarrollo de la solicitud de mejora	• Comité de Control de Cambios • Administrador de Proyecto
Devolver	Devuelve la solución desarrollada para la solicitud de mejora por no pasar satisfactoriamente las pruebas	• Tester • Administrador de Proyecto
Reasignar	Permite reasignar la solicitud de mejora	• Comité de Control de Cambios • Administrador de

		Proyecto
Reabrir	Permite volver a abrir una solicitud que ya ha sido cerrada por los siguientes motivos: a) Ya fue liberada a producción b) Fue devuelta al no pasar las pruebas respectivas c) Porque se indicó que su implementación estaba lista pero se requiere agregar más actividades de trabajo	Según en caso: a) Comité de Control de Cambios b) Administrador de Proyecto c) Administrador de Proyecto
Probar	Indica que la mejora desarrollada está siendo validada	• Tester • Coordinador de Cambios
Confirmar Pruebas	Indica que las pruebas han culminado satisfactoriamente.	• Tester • Coordinador de Cambios
Repetir Pruebas	Solicita que las pruebas aplicadas sobre la solución desarrollada sean repetidas.	• Comité de Control de Cambios • Administrador de Proyecto • Coordinador de Cambios
Cerrar	Marca la solicitud de mejora como finalizada, es decir que ya fue atendida e implementada al 100% en ambiente de producción.	• Coordinador de Cambios • Comité de Control de Cambios • Administrador de Proyecto

Tabla 12: Acciones y roles para solicitudes de mejora

4.5.4 Comportamientos

Es de vital importancia definir los comportamientos que puede tener cada campo de las solicitudes de mejora receptadas (detallados en la sección 4.5.1) en un estado determinado, puesto que permiten proteger las modificaciones de valores en un punto específico del proceso en el cual no sería óptimo hacerlo.

A continuación se presenta una matriz de posibilidades donde se indica si el cambio es opcional (O), mandatorio (M), o si únicamente es permisible de lectura ( ).

Campo \ Estado	Enviada	Duplicada	Asignada	Abierta	Rechazada	Pospuesta	Implementada	En Pruebas	Devuelta	Probada	Cerrada
Cliente	M										
Aplicativo	M										
Descripción	M										
Archivos	O	O	O								
Solicitante	M										
Email contacto	M	O	O	O	O	O	O	O	O	O	O
Estado	M	M	M	M	M	M	M	M	M	M	M
Fecha de envío	M										
Fecha de inicio		M		M	M						
Fecha de cierre		M			M	M					M
Duración		M			M	M					M
Fecha de integración											M
ID Caso			M								
Receptor	M										
Coordinador de Cambios			M								

Administrador de Proyecto			M								
Prioridad			M								
Actividades			M								
Desarrollador			M								
Tester			M								
Integrador			M								
Artefactos				M							
Script de Integración				M						O	
Observaciones	O	O	O	O	O	O	O	O	O	O	O

Tabla 13: Matriz de comportamientos para solicitudes de mejora

4.6 Administración de Actividades Generales

4.6.1 Campos y estructura

Para cada actividad de trabajo en general que se requiera realizar dentro de un proyecto en marcha, deberán registrarse los siguientes campos:

Campo	Descripción
Cliente	Cliente al que pertenece el proyecto sobre el cual se solicita una actividad.
Descripción	Descripción detallada de la actividad a desarrollar.
Archivos	Archivos que ayuden a clarificar el requerimiento.
Solicitante	Persona que solicita el desarrollo.
Email contacto	Correo electrónico al que debe comunicarse cualquier información con respecto al caso.
Proyecto	Proyecto al que pertenece la actividad solicitada.
Administrador de	Nombre del Administrador del Proyecto encargado de planificar,

Proyecto	desarrollar y dar seguimiento a la actividad solicitada.
Estado	Estado actual de la actividad.
Fecha de envío	Fecha en la que se envía la solicitud.
Fecha de inicio	Fecha en la que se inicia a trabajar en la solución.
Fecha de cierre	Fecha en la que se cierra el caso.
Duración	Tiempo que tomó en desarrollar la actividad.
ID Caso	Secuencial que identifica a la actividad solicitada.
Desarrollador	Personal del equipo de desarrollo encargado de realizar la actividad.
Artefactos	Todos los componentes ya sean modulares o de base de datos que se crearon, modificaron o eliminaron para desarrollar la actividad solicitada, por ejemplo: Módulos, reportes, scripts ejecutados, objetos de bases de datos, etc. Deberán incluir los siguientes datos: Tipo: modulo, reporte, disparador, paquete, etc. Nombre: nombre del artefacto Fecha de actualización: fecha de modificación final
Registros de Pruebas	Descripción del conjunto de pruebas aplicadas para validar que la solución desarrollada cumpla con lo requerido.
Observaciones	Cualquier observación o nota que sea necesario incluir con respecto a la actividad que se está desarrollando, deberá incluir: Personal/Cargo: Nombre del personal que escribe la observación y su cargo Notas: Observación Fecha: Fecha en la que realiza la observación

Tabla 14: Campos de la solicitud para actividades generales

4.6.2 Flujo de trabajo

Los estados por los que puede atravesar una actividad solicitada durante el proceso de su desarrollo se explicarán usando tres diagramas, donde se evidencian los posibles escenarios que pueden presentarse, para ello a continuación se presenta un listado de los estados probables a lo largo del proceso.

Estado	Descripción
Enviada	Estado inicial de la actividad solicitada.
Asignada	Cuando la actividad ha sido asignada a un Desarrollador.
Abierta	Cuando el desarrollador comienza a trabajar la actividad solicitada.
Resuelta	Cuando la implementación de la actividad se ha completado.
Pospuesta	Cuando el desarrollador recibe la orden del Administrador de Proyecto de posponer el trabajo para dedicarse a otra actividad de mayor prioridad.

Tabla 15: Estados posibles de una solicitud de actividades generales

Diagrama Básico. Define los estados y transiciones normales por las que pasaría una actividad, suponiendo un escenario ideal en donde no se registre ningún inconveniente, es decir el flujo básico.

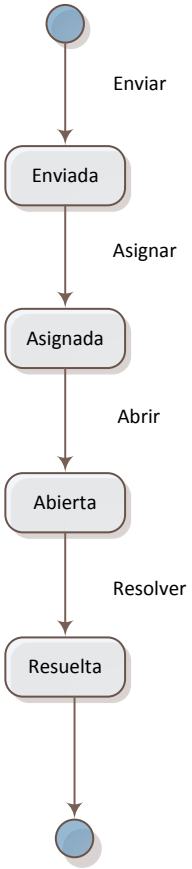


Ilustración 18: Diagrama de flujo básico de una solicitud de actividades generales

Diagrama Completo. Este diagrama modela todas las transiciones del diagrama básico anteriormente definido, más todas las posibles transiciones por las que podría pasar una actividad en casos excepcionales.

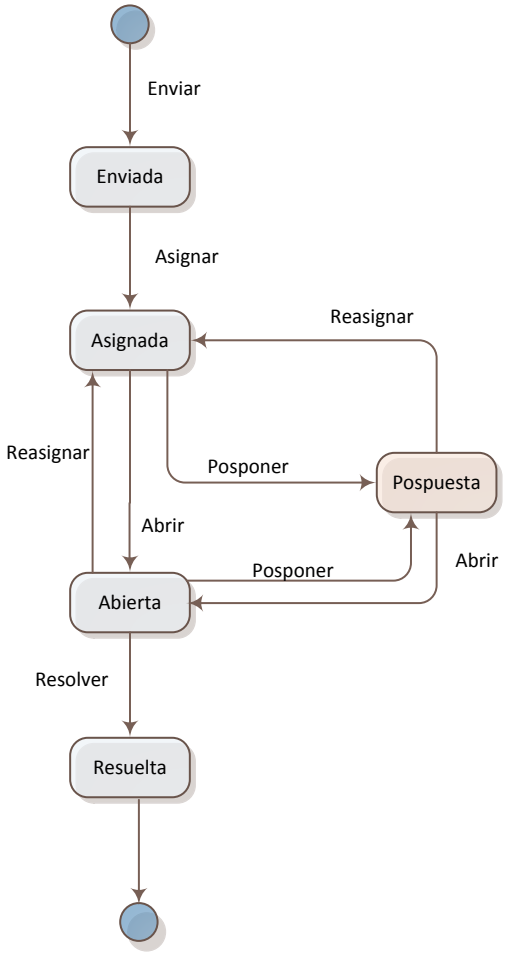


Ilustración 19: Diagrama de flujo completo de una solicitud de actividades generales

El siguiente diagrama muestra un ejemplo para una mejor comprensión del flujo completo de transiciones, de una solicitud de actividades generales.

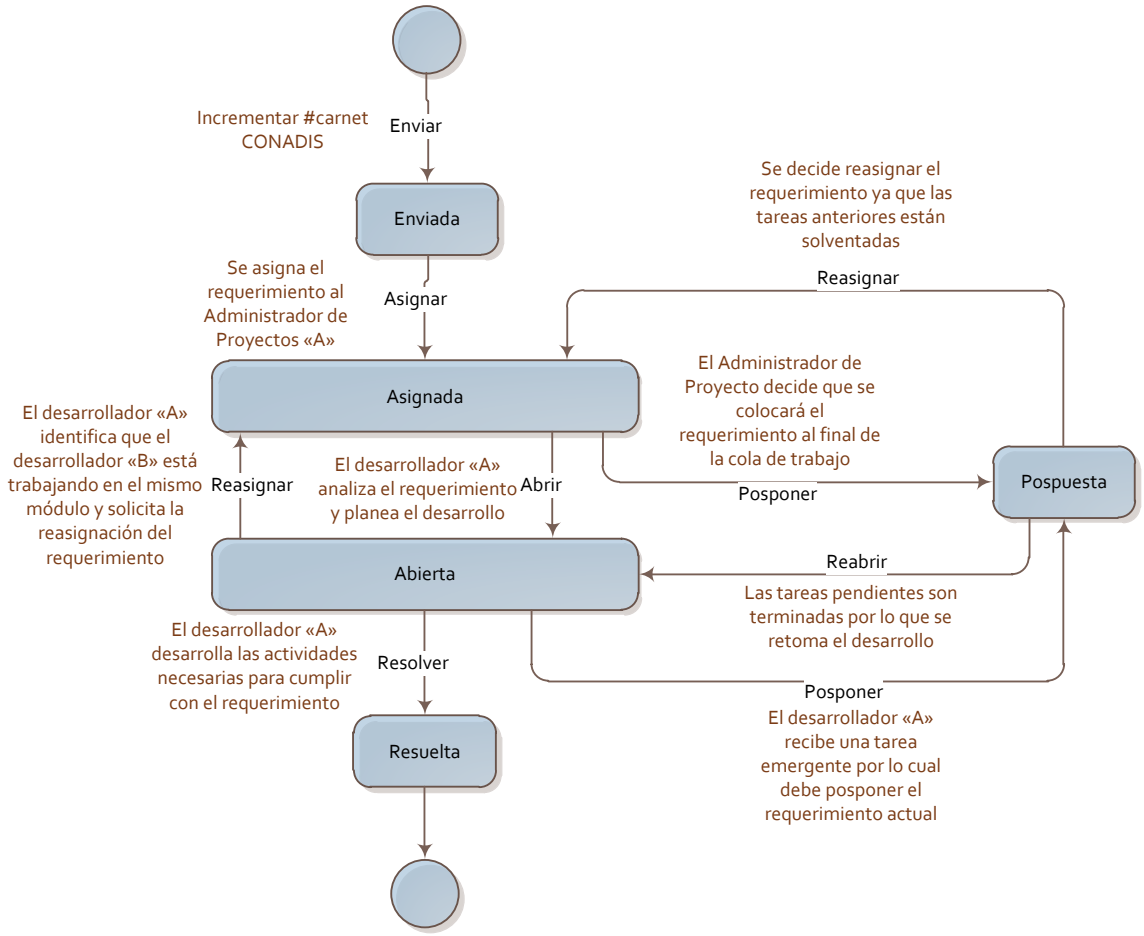


Ilustración 20: Ejemplo de flujo completo de una solicitud de actividades generales

Diagrama de Transiciones por roles. Este diagrama muestra la participación de cada uno de los roles durante el ciclo de desarrollo de una actividad, en todas las posibles transiciones que puedan presentarse.

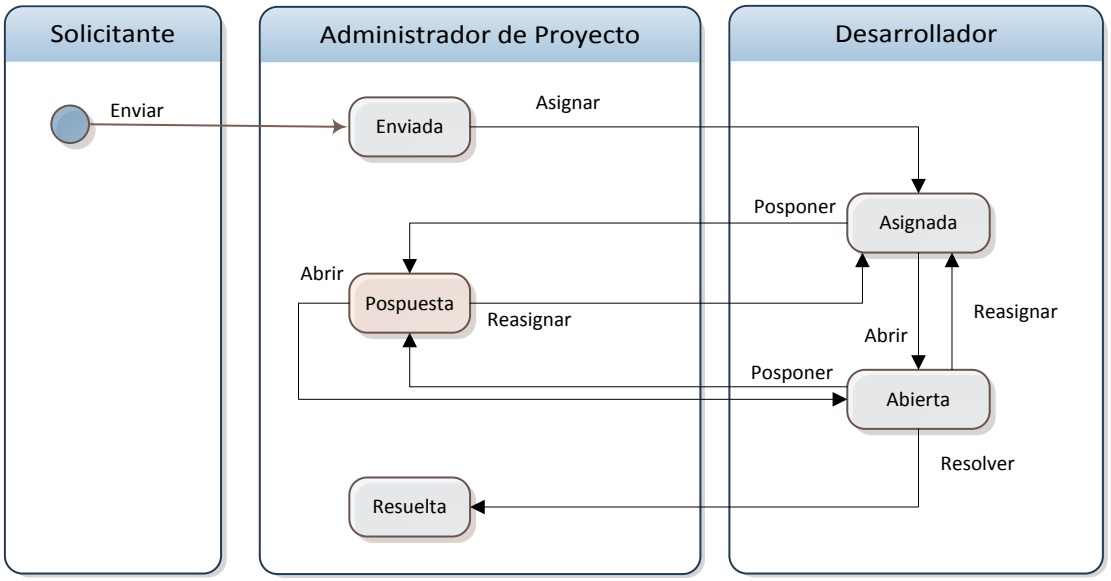


Ilustración 21: Diagrama de transición de actividades por roles para Solicitudes de actividades generales

4.6.3 Acciones

Durante todo el proceso del desarrollo de la actividad solicitada, se presentan distintas acciones las mismas que pueden ser realizadas por ciertos roles específicos en función de sus responsabilidades asignadas.

El siguiente cuadro detalla las acciones que pueden realizarse sobre una actividad y los roles habilitados para ejecutarlas.

Acción	Descripción	Roles Habilitados
Enviar	Envía la solicitud de desarrollo de una actividad	• Todos
Asignar	Asigna la realización de una actividad a un Desarrollador.	• Administrador de Proyecto
Abrir	Marca el inicio de los trabajos para desarrollar la actividad solicitada	• Desarrollador

Resolver	Indica que el desarrollo de la actividad ha concluido.	• Desarrollador
Posponer	Detiene el desarrollo de la actividad.	• Administrador de Proyecto
Reasignar	Reasigna la actividad de trabajo	• Administrador de Proyecto • Desarrollador

Tabla 16: Acciones y roles para solicitudes de actividades generales

4.6.4 Comportamientos

Es de vital importancia definir los comportamientos que puede tener cada campo de las solicitudes de desarrollo de una actividad (detallados en la sección 4.4.1) en un estado determinado, puesto que permiten proteger las modificaciones de valores en un punto específico del proceso en el cual no sería óptimo hacerlo.

A continuación se presenta una matriz de posibilidades donde se indica si el cambio es opcional (O), mandatorio (M), o si únicamente es permisible de lectura ( ).

Estado / Campo	Enviada	Asignada	Abierta	Resuelta	Pospuesta
Cliente	M				
Descripción	M				
Archivos	O	O			
Solicitante	M				
Email contacto	M	O	O	O	O
Proyecto	M				
Administrador de Proyecto		M			
Estado	M	M	M	M	M

Fecha de envío	M				
Fecha de inicio			M		
Fecha de cierre				M	M
Duración				M	M
ID Caso		M			
Desarrollador		M			
Artefactos			M		
Observaciones	O	O	O	O	O

Tabla 17: Matriz de comportamientos para solicitudes de actividades generales

4.7 Capacitación y Recursos

Para conseguir que todo el personal de Softcase logre comprender completamente el enfoque y la importancia de contar con un Plan de Administración de Cambios, sus objetivos y la imperatividad de manejarlo y aplicarlo adecuadamente, sería recomendable aplicar el siguiente esquema de capacitación:

Temas Sugeridos	Personal sugerido
Lectura completa del proceso de Administración de la Configuración y el Plan de Administración de Cambios estandarizado dentro de la empresa. Con esta lectura el personal podrá comprender el detalle de cada una las actividades relacionadas, los roles involucrados y los flujos de proceso de cada tipo de requerimiento.	• Todo el personal
Socialización y capacitación sobre la creación y campos requeridos dentro de los formularios para cada tipo de solicitud.	• Comité de Administración de Cambios • Administradores de

	Proyecto • Coordinadores de Cambio
Resolución de casos prácticos que involucren, correcciones de defectos, solicitudes de mejora y desarrollo de actividades generales dentro de un proyecto en marcha.	• Todo el personal
Análisis del manejo y procedimientos, de cada una de las actividades detalladas para cada tipo de solicitud dentro del Plan de Administración, roles involucrados y acciones permitidas para cada uno en función de sus estados.	• Comité de Administración de Cambios
Análisis del manejo de actividades permitidas para Coordinadores de cambio, en cada tipo de requerimiento y su transición de estados.	• Coordinadores de Cambios
Análisis del manejo de actividades permitidas para Administradores de Proyecto, en cada tipo de requerimiento y su transición de estados.	• Administradores de Proyecto
Análisis del manejo de actividades permitidas para Desarrolladores, en cada tipo de requerimiento y su transición de estados.	• Desarrolladores
Análisis del manejo de actividades permitidas para Testers, en cada tipo de requerimiento y su transición de estados.	• Testers
Análisis del manejo de actividades permitidas para Integradores, en cada tipo de requerimiento y su transición de estados.	• Integradores

Tabla 18: Detalle de capacitación según roles

CAPITULO 5
CONCLUSIONES

5. CONCLUSIONES

5.1 Conclusiones Generales

- Es importante para cualquier empresa desarrolladora de software invertir el tiempo que sea necesario para definir no solo sus procesos sino también su plan de administración de cambios, ya que una vez establecido, los beneficios que se derivan del mismo hacen que valga la pena el tiempo y los recursos que tomó su conformación.
- Sin duda alguna para Softcase, es de vital importancia llevar una correcta Administración de Cambios, ya que sin ella es complicado saber con certeza el funcionamiento exacto de cada uno de los aplicativos instalados en cada cliente, pues para cada uno de ellos el software ha sido personalizado.
- Al organizar y estandarizar las actividades de trabajo y la forma en que se manejan los requerimientos desde su recepción, vuelve posible el hacer un seguimiento de cada solicitud en una línea de tiempo.
- El plan de administración es la forma más práctica y clara de llevar un control en cuanto al qué, quién, cuándo y el porqué de cada modificación realizada sobre el software para cada cliente de Softcase.
- Establecer los roles encargados de cada una de las actividades dentro de los flujos creados para el tratamiento de cada solicitud, evitará duplicidad de trabajo ya que cada persona tiene claramente delimitadas sus responsabilidades y definido el momento oportuno en el que deben ser ejecutadas únicamente las tareas que le son asignadas.
- Los distintos tipos de requerimientos no pueden ser tratados de la misma manera, debe darse un tratamiento específico ya que cada uno de ellos implica un nivel de especificación y especialidad distinta.
- Las herramientas existentes en el mercado brindan un apoyo inminente, puesto que facilitan la creación y seguimiento de los flujos de trabajo establecidos para cada tipo de solicitud; además sus estadísticas al ser automatizadas brindan un

panorama claro, actual e inmediato de diversos aspectos tales como: trabajo que realiza personal de la empresa, clientes que están siendo atendidos, asignaciones de tareas por persona, tiempos de respuesta, entre otros.

5.2 Cumplimiento de objetivos

- Se definieron los procedimientos que seguirá el personal de SoftCase para atender los distintos tipos de requerimiento que se presentan.
- Se estableció el Plan de Administración de Cambios como un artefacto básico dentro del proceso de desarrollo de software de Softcase.
- Se recomendó el uso de la herramienta JIRA de la compañía Atlassian para la posterior automatización del Plan de Administración de cambios por los aspectos citados en el punto 4.3.
- Se describió de forma clara y detallada todo el Plan de Administración de Cambios, sus procesos y flujos de trabajo, de manera que al aplicarlo, todo el personal mantenga un estándar de trabajo y así puedan retomar procesos pendientes en cualquier etapa que estos se encuentren.

5.3 Experiencias

- Mientras se levantaba la información para conformar el Plan de Administración de Cambios, se detectó que en varias ocasiones existió duplicidad de trabajo, por falta de comunicación y por una inminente falta de un proceso claro y establecido del manejo de requerimientos.
- Cuando se desarrollaba este proyecto fue de vital importancia contar con la experiencia de cada una de las personas involucradas, pues se determinaron los escenarios que debían abarcarse y se evidenció que los roles no estaban plenamente establecidos; por ello prácticamente todo el personal realizaba todo tipo de actividades, lo cual mostraba tiempos de respuesta prolongados y producto de ello costos altos. De esta forma la empresa presenta una imagen de

desorganización ante sus clientes, cuyas solicitudes son tratadas de maneras distintas dependiendo de quién recepte el requerimiento.

- Ya que el personal de Softcase estaba acostumbrado a desarrollar varias tareas de diversa índole, es decir no tenían actividades específicas a su cargo, se definieron roles para cada empleado en base a sus habilidades y experiencia.
- Al final de la socialización del Plan de Administración de Cambios, el personal de Softcase se mostró optimista y motivado a dedicarse únicamente a las actividades que les corresponde según sus roles asignados, conscientes de que el resultado de cada actividad, es de vital importancia dentro de todo el proceso para obtener el producto final.

CAPITULO 6

RECOMENDACIONES

6. RECOMENDACIONES

- Definir entregables para cada actividad de manera que se formalice el proceso que deben seguir las solicitudes receptadas.
- Crear un proyecto que defina las métricas que Softcase necesita para evaluar aspectos importantes que aporten a una adecuada toma de decisiones, no solo con respecto a las actividades que realiza el personal, sino en cuanto al manejo de su cartera de clientes.
- Implementar el Plan de Administración de Cambios dentro de JIRA, con lo cual el proceso administrativo y documental de cada caso, será llevado de forma ágil y eficiente, sin mencionar que los reportes requeridos podrán obtenerse de manera inmediata.
- Puesto que el personal de Softcase no contaba con una asignación de rol específico para desarrollar sus actividades previo a la implementación del Plan de Administración de Cambios y que se vio necesario asignarlos en función de su experiencia; se recomienda evaluar su desempeño en un lapso de 3 meses, para analizar la posibilidad de rotar dichos roles y volver a evaluarlos posteriormente, de manera que se optimice sus habilidades.
- Finalmente se recomienda generar un manual de operaciones que abarque todo el proceso, donde se establezca claramente los pasos a seguir desde que un requerimiento es receptado, hasta su respectivo cierre, incluyendo responsables y casos de uso que ejemplifiquen claramente su aplicación; esto servirá de gran ayuda sobre todo en caso que se incluya personal dentro de su nómina.